TAMPEREEN TEKNILLINEN YLIOPISTO

TURKKA MANNILA

MONITOIMIPISTEJÄRJESTELMÄN DATAMIGRAATIO KESKITETTYYN TIETOKANTAAN

Diplomityö

Tarkastaja:
professori Tommi Mikkonen
Tarkastaja ja aihe hyväksytty Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 9. toukokuuta 2012

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
Mannila, Turkka: Monitoimipistejärjestelmän datamigraatio keskitettyyn tietokantaan
Diplomityö, 49 sivua
Kesäkuu 2012
Pääaine: Ohjelmistotuotanto
Tarkastaja: professori Tommi Mikkonen
Avainsanat: datamigraatio, tietojärjestelmäuudistus

Datamigraatio on tietojärjestelmäuudistuksiin liittyvä tehtävä, jossa vanhan järjestelmän data viedään uuteen järjestelmään. Datamigraatiossa tietoa ei pelkästään kopioida uuteen järjestelmään, vaan datan rakenteelle täytyy suorittaa transformaatio kohdetietorakenteisiin.

Tietojärjestelmäuudistuksia tehdään yrityksissä liiketoiminnallisista syistä, jolloin datamigraation toimittamiseen liittyy myös liiketoiminnallisia näkökulmia. Tietojärjestelmäuudistusten tavoitteena on yleensä parantaa yrityksen datan saatavuutta ja laatua. Tämän tavoitteen saavuttamiseksi täytyy lähdesovelluksen dataa analysoida ja tarpeen vaatiessa ehostaa.

Tässä diplomityössä luodaan kokonaiskuva datamigraatioon liittyvistä työvaiheista ja siten autetaan rakentamaan toimiva datamigraatioprosessi tuotannollista datamigraatiota varten. Työssä esitellään Matthesin "Towards an integrated data migration process model" -artikkelissa kokoama datamigraatioprosessimalli ja Atostekin toimittama datamigraatioprojekti. Toiminnallisesta näkökulmasta datamigraatioprojektissa tärkeintä on muodostaa dokumentoitu, testattu ja harjoiteltu datamigraatioprosessi, jota noudattaen datamigraatio suoritetaan tuotannossa. Teknisestä näkökulmasta datamigraatioprojektissa rakennetaan datamigraatioalusta, joka sisältää tarvittavat välitietovarastot sekä datan purku-, transformaatio- ja vientiohjelmat. Lisäksi datamigraatioon liittyvien ohjelmien suoritusten hallitsemista varten voidaan tuottaa erillinen orkestrointikomponentti.

Atostekin toimittamassa datamigraatioprojektissa ei sovellettu tarkoituksella mitään datamigraatioprojekteihin liittyviä viitekehyksiä, mutta projektissa tuotettu datamigraatioprosessi vastasi Matthesin datamigraatioprosessimallia. Atostekin toimittamassa projektissa muodostettiin yksityiskohtaisesti dokumentoitu, testattu ja harjoiteltu datamigraatioprosessi, jonka avulla tuotannollinen datamigraatio suoritettiin onnistuneesti. Datamigraatioprosessin tueksi tuotettiin työkaluja, joita voidaan uudelleen käyttää myös tulevissa datamigraatioprojekteissa.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Technology
Mannila, Turkka: Multi-office system's data migration into centralized database
Master of Science Thesis, 49 pages
June 2012
Major: Software Engineering
Examiner: professor Tommi Mikkonen
Keywords: data migration, information system renewal

Data migration is an endeavor closely related to information system renewals, where old data is moved to the new system. Data migration is not merely the act of copying data, instead data must be also transformed to target data structure.

Information system renewals are executed in corporations because of business reasons. Therefore business aspects play a significant role in data migration. One important goal of data system renewals is to improve availability and quality of corporate data. To achieve this goal source application's data must be analyzed and enhanced if necessary.

This master's thesis creates an overall picture of endeavors related to data migration and therefore helps reader to build a functioning data migration process for production use. This thesis presents the data migration process model composed by Matthes in article "Towards an integrated data migration process model" and a data migration project delivered by Atostek. Primary functional milestone of a data migration project is to create a documented, tested and rehearsed data migration process. Data migration is executed in production by utilizing this process. Primary technical milestone is to build a data migration platform, which includes all required applications and databases. Furthermore, a distinct orchestration component can be produced to govern the execution of data migration programs.

Data migration project delivered by Atostek was not planned based on any data migration framework. However, the data migration process created in the project resembled Matthes's data migration process model. Documented, tested and rehearsed data migration process was produced in Atostek's data migration project. Furthermore, it was used to successfully execute the data migration in production. Supportive tools were also produced, which can be re-used in future data migration projects.

ALKUSANAT

Tämän diplomityön aihe on peräisin Atostekin toimittamasta datamigraatioprojektista. Atostek on myös rahoittanut tämän diplomityön. Kiitokset Atostekille työn aiheesta ja rahoituksesta.

Työn ohjaajina toimivat Atostekilta Mika Torhola ja Tampereen teknilliseltä yliopistolta professori Tommi Mikkonen. Mika toimi projektipäällikkönä työssä esitetyssä datamigraatioprojektissa. Haluan kiittää Mikaa loistavasta ohjaustyöstä ja työn sisältöä koskevista neuvoista. Tommia haluan kiittää loistavasta ohjaustyöstä ja hyvistä kommenteista. Lisäksi Tommin nopeat vastaukset edesauttoivat työni sujuvaa etenemistä.

Kotiväkeä haluan kiittää tuesta diplomityön kirjoituksen aikana. Erityisesti kiitokset puolisolleni Marisalle, joka auttoi myös työni oikolukemisessa.

Tampereella 18.5.2012

Turkka Mannila

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

Asiakas	Datamigraation toimituksen tilannut osapuoli.
Datamigraatio	Ainutkertainen työkalu-tuettu prosessi, jonka päämääränä on muuttaa dataa lähdetietorakenteista kohdetietorakenteisiin, missä molemmat rakenteet eroavat käsitteellisellä ja/tai fyysisellä tasolla.
Datamigraatioikkuna	Ajanjakso, jonka aikana datamigraation kohteena oleva järjestelmä on suunnitelmallisesti pois käytöstä.
Datamigraatiokäsikirjoitus	Datamigraatioprosessin suorituksen kuvaava dokumentti.
Datamigraatioprojekti	Datamigraatioprojektin aikana luodaan datamigraatioprosessi ja suoritetaan se tuotannossa.
Datamigraatioprosessi	Joukko peräkkäisiä ohjelmien tai ihmisten suorittamia tehtäviä, joiden suorittamisen jälkeen data on viety lähdejärjestelmästä kohdejärjestelmään.
Datan laatu	Kuinka hyvin data vastaa sen kuvaamaa todellisen maailman ilmiötä.
Datan purkusovellus	Datan purkusovellus purkaa datan lähdesovelluksesta lähdekerääntymisalueelle.
Datan transformaatio	Datan rakenteellinen tai muotoilullinen muunnos lähdetietorakenteesta kohdetietorakenteeseen.
Datan vientisovellus	Datan vientisovellus vie datan kohdesovellukseen kohdekerääntymisalueelta.
Kohdekerääntymisalue	Transformoitu data siirretään kohdekerääntymisalueelle datan transformaation jälkeen odottamaan vientiä kohdesovellukseen.
Kohdesovellus	Kohdesovellukseen viedään lähdesovelluksen data.
Lähdekerääntymisalue	Data viedään lähdesovelluksesta lähdekerääntymisalueelle datan transformaatiota varten.
Lähdesovellus	Sovellus, josta data on tarkoitus viedä kohdesovellukseen.
Orkestrointikomponentti	Hallitsee datamigraatioprosessin suoritukseen liittyvien sovellusten suoritusjärjestystä ja keskinäistä toimintaa.
Transformaatiosovellus	Datan transformaation suorittava sovellus.

1 JOHDANTO

Datamigraatio on tietojärjestelmäuudistuksiin liittyvä tehtävä, jossa vanhan järjestelmän data viedään uuteen järjestelmään. Data on nykyään arvokasta yrityksen omaisuutta, jonka saatavuutta ja laatua pyritään parantamaan uudistamalla tietojärjestelmiä. Datamigraatio ei ole pelkkää tiedon kopiointia, vaan siihen liittyy myös datan transformaatio ja datan laadun ehostus kohdejärjestelmän vaatimusten mukaiseksi. Lisäksi datamigraatiossa minimoidaan järjestelmän käyttökatkosta asiakkaalle aiheutuva haitta.

J. Morris toteaa [1 s. 15], ettei datamigraatio ole ensisijaisesti tekninen haaste, vaan se on liiketoiminnallisen osaamisen haaste. Datamigraatioprojektin toimittaminen on osaksi konsultointityötä, jonka tavoitteena on auttaa asiakasta saavuttamaan järjestelmäuudistuksen tavoitteet. Datamigraatioprojektissa täytyy myös löytää järjestelmäuudistukseen liittyvät tietovarastot ja dokumentoida ne. Lisäksi projektissa täytyy tunnistaa datan sidosryhmät, joilta saadaan arvokasta datamigraatioprojektin läpivientiä helpottavaa tietoa.

Datamigraatiota on pääsiassa kahta eri tyyppiä [2]: *datavaraston migraatio* eli käytännössä tietokannan tai palvelimen muutto tai päivitys. Datavaraston migraatioon sisältyy vähän tai ei ollenkaan datan rakenteellisia muutoksia. Toinen tyyppi on *sovellusdatan migraatio,* joka sisältää tiedon siirtämisen lisäksi myös tiedon rakenteen muokkaamisen uuteen järjestelmään sopivaksi. Tämä diplomityö keskittyy edellisen jaottelun perusteella sovellusdatan migraatioon, joka on huomattavasti datavaraston migraatiota työläämpää. Datamigraatio-termi määritellään siis tässä diplomityössä seuraavasti [3 s. 6]: *"Ainutkertainen työkalu-tuettu prosessi, jonka päämääränä on muuttaa dataa lähdetietorakenteista kohdetietorakenteisiin, missä molemmat rakenteet eroavat käsitteellisellä ja/tai fyysisellä tasolla".* Tämä määritelmä sisältää sanan "prosessi", jonka luominen on datamigraatioprojektin tehtävä. Datamigraatio suoritetaan tuotannossa datamigraatioprosessin mukaisesti.

Datamigraatioprojektit epäonnistuvat Gartnerin mukaan useammin kuin muut IT-projektit. Datamigraatioprojektin epäonnistuminen tarkoittaa myös järjestelmäuudistuksen epäonnistumista, sillä uuden järjestelmän toiminta usein riippuu vanhan järjestelmän liiketoiminta-datasta. Datamigraatio järjestelmäuudistuksen yhteydessä ajatellaan usein virheellisesti olevan vain tiedon kopiointia vanhasta järjestelmästä uuteen. Siksi datamigraation osuus liiketoimintasovelluksen toteutus- ja käyttöönottoprojekteissa alimitoitetaan.

Datan laadun analyysi ja tarvittaessa ehostus liittyy myös datamigraatioprojektin tehtäviin. Järjestelmäuudistuksen tavoitteena on usein parantaa yrityksen datan saatavuutta ja laatua. Tällöin pelkkä vanhan järjestelmän datan vieminen sellaisenaan uuteen

järjestelmään ei riitä, vaan datan laatu täytyy ehostaa uuden järjestelmän vaatimusten mukaiseksi.

Tässä diplomityössä esitellään datamigraatioon liittyvät tehtävät korkealla tasolla, jolloin se osataan ottaa oikeassa laajuudessa huomioon erillisenä työkokonaisuutena uuden järjestelmän käyttöönottoprojektin yhteydessä. Tarkoitus ei ole esittää universaalia ratkaisua datamigraatio-ongelmaan, vaan viitekehys, jonka ympärille ratkaisu voidaan rakentaa. Viitekehyksenä toimii F. Matthesin [3] kokoama datamigraatioprosessimalli. Viitekehys määrittelee joukon tehtäviä, joiden olemassaolon ymmärtäminen on lähtökohta onnistuneelle datamigraatioprojektille. Lisäksi työssä esitellään muutamia näiden tehtävien suorittamista helpottavia työkaluja.

Työssä esitellään myös Atostekin toimittama datamigraatioprojekti. Projektin läpivientiä ei suunniteltu minkään datamigraatioon liittyvän viitekehyksen mukaan, mutta se vietiin onnistuneesti läpi. Atostekin toimittamassa datamigraatiossa luotiin dokumentoitu, testattu ja harjoiteltu datamigraatioprosessi, jota noudattaen datamigraatio suoritettiin tuotannossa.

Luvussa 2 esitetään syitä tietojärjestelmäuudistuksille, datamigraatio liiketoimintanäkökulmasta, datamigraatioprojektin yleiskuvaus ja datamigraatioarkkitehtuuri teknisestä näkökulmasta. Luvussa 3 esitellään F. Matthesin kokoama datamigraatioprosessimalli. Datamigraatioprosessimallin vaiheita täydennetään myös muiden esittämillä teorioilla. Luvussa 4 esitellään Atostekin toimittama datamigraatioprojekti. Luvussa 5 arvioidaan sekä teoriasta että käytännöstä opittuja asioita. Lisäksi otetaan kantaa datamigraatioon palvelutuotteena. Luvussa 6 tiivistetään tärkeimmät diplomityön aikana opitut asiat.

2 DATAMIGRAATIO

Datamigraatio on tietojärjestelmäuudistukseen liittyvä tehtävä, jossa vanhan järjestelmän data viedään uuteen järjestelmään. Tässä luvussa esitetään syitä tietojärjestelmäuudistuksille, datamigraatio liiketoimintanäkökulmasta, datamigraatioprojektin yleiskuvaus ja datamigraatioarkkitehtuuri teknisestä näkökulmasta.

2.1 Syitä tietojärjestelmäuudistukselle

Tietojärjestelmäuudistuksia ei tehdä niihin liittyvien riskien ja kustannusten vuoksi kevein perustein, mutta seuraavat seikat ajavat yrityksen tilanteeseen, jossa ei ole muuta mahdollisuutta [3 s. 2]:

- **Tiedon integrointi ja lujittaminen** vaatii muutoksia tietojärjestelmään (esimerkiksi organisaatioiden yhdistyminen).
- **Uudet lait ja säädökset** aiheuttavat muutoksia liiketoimintaprosesseille, jotka taas aiheuttavat muutoksia liiketoimintasovelluksille ja tiedon rakenteelle. Esimerkiksi EU direktiivi INSPIRE [4] Euroopan yhteisön paikkatietoinfrastruktuurin perustamisesta ajaa useita julkisia organisaatioita uudistamaan tietojärjestelmiään.
- **Uudet liiketoimintamallit ja prosessit** aiheuttavat uusia toiminnallisia ja eitoiminnallisia vaatimuksia nykyisille sovelluksille. Tällöin vaihtaminen uuteen järjestelmään on usein edullisempaa kuin vanhan muokkaaminen.
- **Vanhojen sovellusten ikääntymisestä** seuraa, että niistä tulee vaikeasti ylläpidettäviä. Sovellusten rakenteella ja dokumentaatiolla on taipumus haurastua ajan kuluessa [5]. Siten vanhojen järjestelmien ylläpito tulee yhä kalliimmaksi ja jossain vaiheessa on kannattavampaa luopua vanhasta ja ottaa käyttöön uusi järjestelmä.
- **Vanhan järjestelmän osaamisen vähenemisestä työmarkkinoilla** seuraa se että, vanhan järjestelmän käyttöä osaavan työvoiman hankkiminen muuttuu vaikeammaksi. Tällöin tulee tarve uudistaa tietojärjestelmä.

Tietojärjestelmän uudistamisen yhteydessä yleensä täytyy vanhan järjestelmän datat viedä uuteen järjestelmään. Muitakin vaihtoehtoja kuitenkin on (esim. tietokanta-tason kääriminen), mutta niitä ei käsitellä tässä diplomityössä.

2.2 Datamigraatio liiketoimintanäkökulmasta

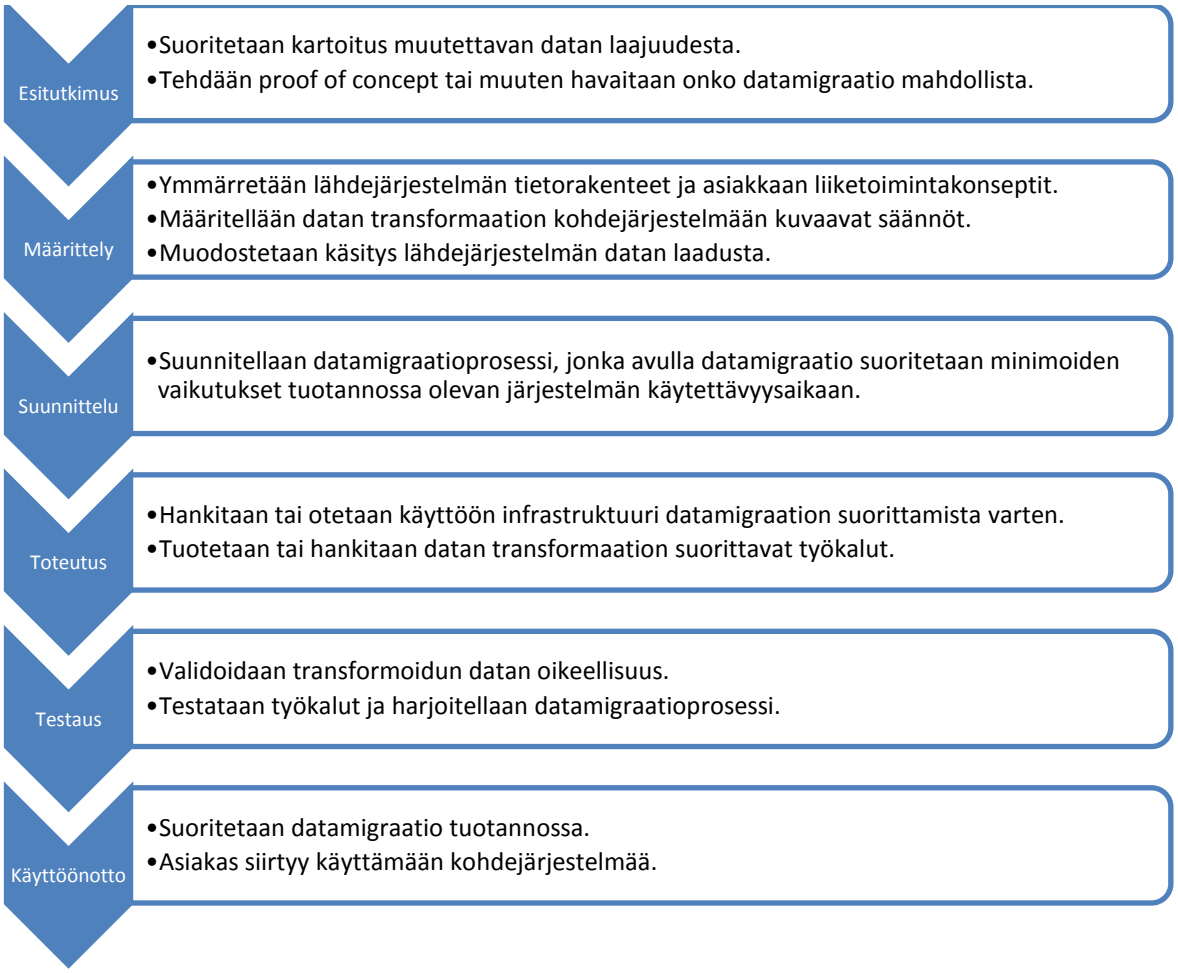
Liiketoimintanäkökulmasta datamigraatioprojekteilla on erityisiä ominaisuuksia [3 s. 7]:

- **Suoritetaan harvoin, mutta suoritetaan väistämättä:** Datamigraatioprojekteja ei tehdä yrityksessä prosessimaisesti, vaan ne ovat luonteeltaan ainutkertaisia tapahtumia. Näin datamigraatioprojekteista ei pääse kertymään osaamista vaan edellisessä projektissa opitut asiat unohtuvat ajan myötä.
- **Riskialtis:** Yrityksen liiketoimintadata on arvokasta immateriaaliomaisuutta, jonka muuntaminen muodosta toiseen on riskialtista [6]. Datamigraation suorittamisen jälkeen vanhaan järjestelmään on hyvin kallista tai mahdotonta palata.
- **Ajaton:** Teknologian kehittyminen ja järjestelmien vaatimusten muuttuminen saavat aikaan sen, että vanhoja järjestelmiä korvataan uusilla.
- **Tiukasti sidottu liiketoimintasovelluksen käyttöönottoprojektiin:** Datamigraatioprojektin onnistuminen vaikuttaa liiketoimintasovelluksen käyttöönottoprojektin onnistumiseen ja toisin päin. Uutta järjestelmää on mahdotonta ottaa käyttöön, ellei datamigraatioprojekti ole onnistunut muuttamaan vanhaa dataa uuteen järjestelmään. Toisaalta taas käyttöönottoprojektin muiden vaiheiden viivästyminen viivästyttää datamigraatioprojektia.

Kaikki nämä ominaisuudet vaikuttavat siihen, että datamigraatio-osaamisesta syntyy asiantuntijayritykselle liiketoimintaprospekteja. Prospektien syntyminen johtuu siitä, että ongelma-aluetta tuntemattoman organisaation ei kannata rakentaa kompetenssia yhtä harvoin läpivietävää projektityyppiä varten. Sen sijaan datamigraatiota tarvitsevalle osapuolelle on paljon edullisempaa vuokrata tarvittava osaaminen ulkopuoliselta toimijalta. [7]

2.3 Datamigraatioprojekti

Datamigraatioprojektin päämäärä on luoda datamigraatioprosessi, jonka avulla lähdejärjestelmän data viedään uuteen järjestelmään. Datamigraatioprojekti sisältää joukon tehtäviä, jotka ovat tunnistettavissa jokaisesta ei-triviaalista datamigraatioprojektista. Kuvassa 2.1 on listattu datamigraatioprojektiin liittyvät tehtävät jaoteltuna esitutkimus, määrittely, suunnittelu, toteutus, testaus ja käyttöönotto -vaiheisiin. Kuvassa esitettyjen vaiheiden suorittaminen hallitusti vaatii suunnittelua. Suunnittelun apuvälineenä voidaan käyttää myöhemmin tässä työssä esiteltävää datamigraatioprosessimallia. Datamigraatioprosessimallin avulla luodaan datamigraation tuotannollisen suorituksen kuvaava datamigraatioprosessi.

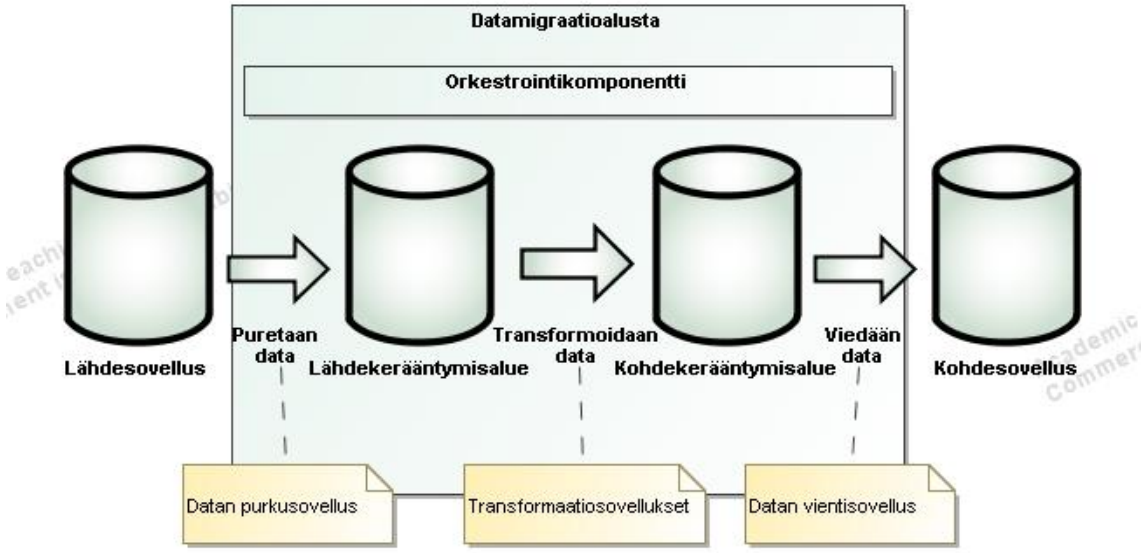


Kuva 2.1. Datamigraatioprojektin tehtävät.

2.4 Geneerinen datamigraatioarkkitehtuuri

Geneerinen datamigraatioarkkitehtuuri kuvaa sen joukon sovelluksia ja tietokantoja korkealla tasolla, jotka luodaan datamigraatioprojektin aikana. Arkkitehtuuriin liittyy keskeisesti datamigraatioalusta (data migration platform), jonka tehtävänä on toimia infrastruktuurina datamigraatioprosessille (Kuva 2.2). Datamigraatioalusta sisältää korkealla tasolla ilmaistuna seuraavat ohjelmistot ja tietovarastot: datan purkusovellus, lähdekerääntymisalue, transformaatiosovellukset, kohdekerääntymisalue ja datan vientisovellus.

Datamigraatioarkkitehtuuri muistuttaa ETL-prosessia (Extract, Transform and Load process), mutta eroaa siitä kuitenkin oleellisesti. Datamigraatio on kertaluontoinen ja siinä transformoidaan vanhan liiketoimintasovelluksen data uuteen liiketoimintasovellukseen. ETL-prosesseja puolestaan käytetään tietyn data-osajoukon transformaatioon ja vientiin reaaliaikaisista liiketoimintasovelluksista raportointisovelluksiin.



Kuva 2.2. Datamigraatioarkkitehtuuri.

2.4.1 Datamigraation tietovarastot

Datamigraatioprosessia varten perustetaan kaksi tietovarastoa: lähdekerääntymisalue (source staging area) ja kohdekerääntymisalue (target staging area). Taulukossa 2.1 on esitettynä datamigraatioprosessin tietovarastojen kuvaukset. [7][3 s. 29]

Taulukko 2.1. Datamigraatioprosessin tietovarastot.

Tietovarasto	Tarkoitus
Lähdesovellus	Lähdesovellus on mahdollisesti tuotantokäytössä oleva sovellus, joka on tarkoitus korvata kohdesovelluksella.
Lähdekerääntymisalue	Lähdekerääntymisalueen tarkoitus on erottaa tuotantokäytössä oleva sovellus, jotta transformaatiosovellusten kehitys on mahdollista häiritsemättä lähdejärjestelmän toimintaa.
Kohdekerääntymisalue	Kohdekerääntymisalueen tehtävä on kerätä transformoitua dataa, joka on valmiina ladattavaksi kohdesovellukseen.
Kohdesovellus	Kohdesovellus on lähdesovelluksen korvaava sovellus, joka on tuotantokäytössä datamigraatioprosessin päätteeksi.

Tietovarastojen välillä suoritetaan seuraavat operaatiot: datan purkaminen (Extract data), datan transformaatio (transform data) ja tiedon vienti (load data). Lisäksi kunkin operaation yhteydessä voidaan suorittaa datan rikastamista (data enrichment), datan validointia (data validation) tai datan suodatusta (data filtering).

2.4.2 Datamigraatioon liittyvät sovellukset

Datamigraatiota varten kehitetään seuraavat sovellukset: datan purkusovellus (data extraction program), transformaatiosovellukset (transformation program) ja datan vientisovellus (data loading program). Näiden sovellusten suoritusjärjestystä hallitsee erillinen orkestrointikomponentti (Orchestration component).

Datan purkusovelluksen tehtävä on louhia datamigraatiossa tarvittava data lähdesovelluksesta lähdekerääntymisalueelle, eli toisin sanoen eriyttää lähdesovelluksen data migraatioprosessin kehityskäyttöön. Purkusovellus voi lisäksi osallistua datan suodattamiseen esimerkiksi jättämällä pois lähdesovelluksen tauluja, sarakkeita tai rivejä.

Transformaatiosovellusten tarkoitus on muuttaa tiedon rakenne kohdejärjestelmään sopivaksi. Tiedon rakenteelliset muutokset voivat sisältää relaatioiden yhdistämisiä, pilkkomisia, lisäämisiä ja poistamisia.

Datan vientisovelluksen tarkoitus on viedä transformoitu data kohdejärjestelmään joko suoraan tietokantaan tai käyttäen kohdesovelluksen tarjoamia datan vientirajapintoja (data loading interface).

Orkestrointikomponentin tehtävä on hallita sovellusten oikeaa suoritusjärjestystä käynnistelemällä datamigraatioon liittyviä ohjelmia ja välittämällä dataa niiden välillä.

2.5 Datamigraation osaamisesta ja tutkimuksesta

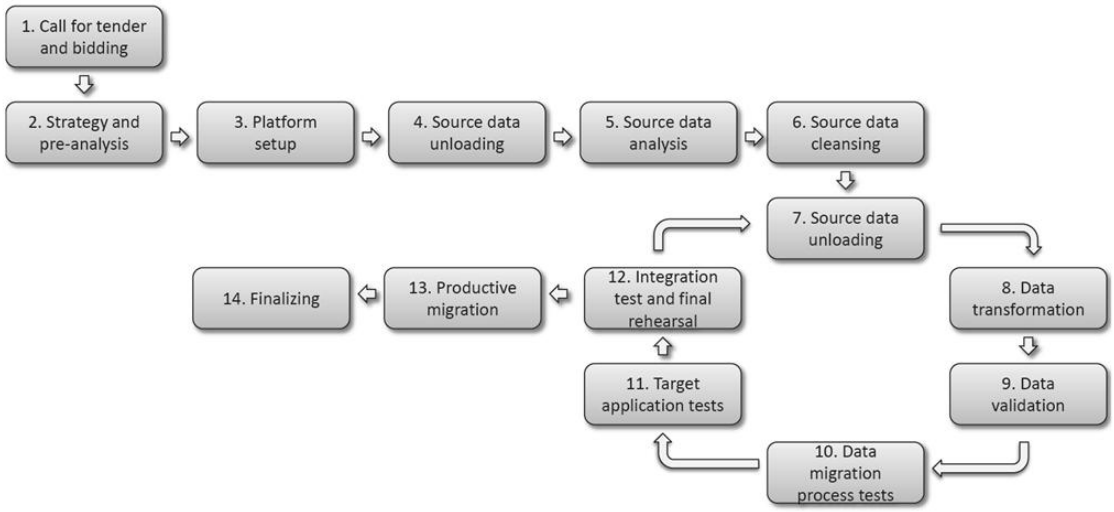
Gartner-ryhmän mukaan "*83 % datamigraatioista epäonnistuu kokonaisuudessaan tai ylittävät niille mitoitetut budjetit ja aikataulut*". Tätä väitettä tukee myös Bloor Research joka on kyselyjen perusteella havainnut, että vain 16 % datamigraatioprojekteista toimitetaan ajallaan ja pysyen budjetissa [8]. Endava puolestaan väittää, että *"vain harvoilla yrityksillä on tarpeelliset taidot johtaa, rakentaa ja implementoida onnistunut datamigraatio"*. Nämä tilastot ovat synkkiä verrattuna siihen, että keskimäärin jopa noin puolet IT-projekteista onnistuu [9]. Tilastoista päätellen datamigraatio-osaamista on saatavilla markkinoilla selvästi vähemmän suhteessa muuhun IT-osaamiseen.

Datamigraatioprojektien tutkimusta on tehty hyvin vähän yrityksissä, tutkimuslaitoksissa ja standardointiorganisaatioissa. Tästä johtuen koko datamigraatioprojektiin kantaa ottavia artikkeleita onkin vähän. Artikkelit yleensä käsittelevät vain jotain yksittäistä datamigraatioprojektin näkökulmaa, kuten laadunhallintaa, resursointia tai datan profilointia. Lisäksi suurin osa artikkeleista käsittelee jonkin tietyn järjestelmän migraatiota versiosta A versioon B tai vaihtoehtoisesti jonkin standardisovelluksen korvaamista toisella. Poikkeuksellisesti Münchenin teknillisen yliopiston SEBIS (Software Engineering for Business Information Systems) -yksikössä F. Matthes tutkii integroitua datamigraatioprosessimallia [3]. Datamigraatioprosessimalli määrittelee tavan rakentaa järjestelmällinen datamigraatioprosessi. Kyseisen tutkimuksen kokoamaan prosessimalliin viitataan myöhemmin tässä työssä laajalti.

Datamigraatioprojektin läpivientiä kuvaavia kirjoja on myös vähän. Tässä työssä viitataan useasti J. Morriksen Practical Data Migration-kirjaan [1]. Kirja esittää datamigraatioprojektin läpiviennin vaihe vaiheelta lukuisten käytännön esimerkkien avulla.

3 DATAMIGRAATIOPROSESSIMALLI

Datamigraatioprosessimalli sisältää 14 vaihetta (Kuva 3.1), joista osa on iteratiivisia (vaiheet 7-12). Lisäksi näiden vaiheiden rinnalla tehdään jatkuvasti projektinhallintaa. Mallin tarkoitus on kuvata sääntöjä ja menetelmiä, joita noudattamalla datamigraatioprojekti voidaan viedä haluttuun tavoitteeseen. Malli ei ole jäykkä. Datamigraatioprosessimallista voidaan jättää pois vaiheita, joita ei katsota tarpeellisiksi projektin läpiviemiseksi. Malli painottaa myös jatkuvaa testaamista koko prosessin aikana. Testaamisella hallitaan datamigraatioon liittyviä riskejä. [3]



Kuva 3.1. Datamigraatioprosessimalli. [3 s. 18]

3.1 Kutsu tarjouspyynnöille

Kutsu tarjouspyynnöille (call for tender and bidding) on vaihe jossa datamigraatiopalvelua tarvitseva osapuoli (asiakas) ilmoittaa tarpeestaan markkinoille, johon datamigraatiopalvelua tarjoava osapuoli vastaa tuottamalla tarjouksen datamigraation toimittamisesta. Järkevän tarjouksen tuottaminen datamigraatiotyöstä vaatii yleiskatsauksen muutettavan datan laajuudesta, laadusta ja liiketoiminnalle aihetuvista riskeistä. Vaiheen lopputuloksena on siis asiakkaalle luovutettava tarjousdokumentti sekä alustava ymmärrys lähdejärjestelmän ja muutettavan datan ominaisuuksista. [3 ss. 18-19]

3.2 Strategia ja esianalyysi

Strategia- ja esianalyysivaiheen (strategy and pre-analysis) tavoitteena on rajata muutettava data, valita lähestymistapa projektiin, määrittää projektiorganisaatio ja datamigraa-

tion rajoitteet sekä teknisistä että liiketoiminnallisista näkökulmista. Lisäksi on hankittava tietoa muutettavasta datasta. Näiden tietojen hankkimiseksi on löydettävä sopivat henkilöt, jotka kykenevät vastaamaan dataa koskeviin kysymyksiin. Näitä henkilöitä kutsutaan datan sidosryhmien edustajiksi.

3.2.1 Datan sidosryhmien edustajat

Datan sidosryhmän edustaja on asiakkaan organisaatiossa tai sen ulkopuolella oleva henkilö, jolla on aito kiinnostus datamigraation lopputuloksiin. Seuraavaksi esitellään muutamia yleisimpiä datan sidosryhmien edustajia. Kaksi ensimmäistä ovat tärkeitä rooleja, jotka on täytettävä. Loput roolit on hyvä olla tiedossa, mutta ne eivät ole yhtä välttämättömiä. [1 ss. 25-37]

Tietovaraston omistaja (data store owner) on rooli, jolla on päätäntävaltuus tietovaraston toiminnasta. Hän voi esimerkiksi sanoa, ajetaanko tietovarasto alas huoltoa varten. Näiden henkilöiden löytäminen voi olla vaikeaa, sillä joitain järjestelmiä käyttävät kaikki, mutta kukaan ei omista niitä. Tietovaraston omistaja on tärkeä voimavara datamigraatioprojektille, sillä hän voi tietää tietovarastolle tehdyistä korjauksista tai siinä esiintyneistä vioista.

Liiketoiminnan toimialueen asiantuntija (business domain expert) on rooli, joka tietää miten tietovarastoa käytetään tällä hetkellä ja joka tuntee sen ominaispiirteet sekä käytössä olevat kiertotiet nykyisiin ongelmiin. Ei siis kannata tyytyä harjoittelijaan, jonka asiakas mielellään nimittää asiantuntijaksi, vaan liiketoiminnan asiantuntijan täytyy olla henkilö, joka todella tuntee järjestelmän ominaisuudet läpikotaisin tai osaa osoittaa ne tuntevan henkilön.

Teknisen tiedon asiantuntija (technical data expert) tuntee tietovaraston teknisestä näkökulmasta, eli hän tuntee tiedostojen formaatit, käyttöoikeudet, rajapinnat ja niin edelleen. On hyvä huomata, ettei järjestelmänvalvoja tunne järjestelmää samalla tavalla kuin järjestelmän käyttäjä. Järjestelmän käyttäjä tuntee järjestelmän liiketoimintanäkökulmasta ja tekninen asiantuntija teknisestä näkökulmasta, joten käyttäjä ei ole paras henkilö tekniseksi asiantuntijaksi ja järjestelmänvalvoja ei ole paras henkilö liiketoiminnan toimialueen asiantuntijaksi.

Ohjelmistoasiantuntija (programme expert) tuntee yleisesti ohjelmistojen toiminnan teknisellä tasolla. Ohjelmistoasiantuntija voi olla sama henkilö kuin datamigraatioasiantuntija. Näitä tehtäviä ei kuitenkaan tule sekoittaa vaan datamigraatioasiantuntijan tulee olla välittäjänä ohjelmistojen, liiketoiminnan ja muiden tiedon sidosryhmien tarpeiden välillä.

Yhtiön data-arkkitehti (corporate data architect) on vastuussa organisaation datan varastoinnin suunnittelusta. Mikäli asiakkaan organisaatiossa on tällainen henkilö, niin se saatetaan yleensä välittömästi datamigraatiota suorittavan osapuolen tietoisuuteen. Näiden henkilöiden päivittäiseen työnkuvaan kuluu tietomallien ylläpito ja datan laadun tarkkailu. Tällöin on ilmeisen selvää, että kyseisistä henkilöistä on hyötyä datamigraatioprojektille. Heiltä voidaan saada legacy-järjestelmän tietomalleja ja arvokasta datan laatuun liittyvää tietoa. Täytyy kuitenkin muistaa, että he eivät kuitenkaan välttämättä

tunne järjestelmän päivittäiseen käyttöön liittyviä näkökulmia samoin kuin liiketoiminnan toimialueen asiantuntijat.

Muut sidosryhmät ovat hyödyksi, jos heillä on jotain arvokasta tietoa dataan ja sen käyttöön liittyen. Täytyy kuitenkin muistaa määritellä heidän roolinsa ja luovuttamien tietojen painoarvo, ettei heidän vaikutusvalta pääse ylittämään heidän todellista kompetenssia.

3.2.2 Legacy-tietovarastot

Legacy-tietovarasto on mikä tahansa tietosäiliö (myös ei-digitaaliset), jossa on mahdollisesti kohdejärjestelmää kiinnostavaa dataa. Nämä tietovarastot täytyy löytää ja dokumentoida, jotta ne voidaan käsitellä systemaattisesti projektin myöhemmissä vaiheissa. Asiakkaalta saadaan mahdollisesti lista tietovarastoista, mutta näitä tietovarastoja voidaan odottaa löytyvän lisää. Tässä vaiheessa projektia kuitenkin on parasta löytää mahdollisimman paljon tietovarastoja, ettei projektin myöhemmissä vaiheissa pääse syntymään yllätyksiä.

Taulukon 3.1 esittämät tiedot olisi hyvä dokumentoida jokaisesta tietovarastosta. [1 ss. 46-47]

Taulukko 3.1. Tietovarastoista dokumentoitavat tiedot.

Tieto	Esimerkki
Tietovaraston nimi	Varastonhallintajärjestelmän tietokanta
Liiketoiminnan alue, johon se liittyy	Varaston materiaalivirran hallinta
Tietovaraston omistaja	Tehtaanjohtaja Heikki Pikkarainen
Liiketoiminnan toimialueen asiantuntija	Työnjohtaja Sakari Rönkkö
Missä tietovarasto fyysisesti sijaitsee	Nakkilan tehtaan palvelinhuone (huone B124)
Muut tiedon sidosryhmät, mikäli tarpeellista	Varastomies Jukka Mietonen, käyttää järjestelmää
Tietovaraston formaatti	Microsoft SQL Server 2008 R2 -tietokanta
Tiedon määrä	15 GB, noin 2 000 000 tietokantariviä
Ylimalkainen käsitys datan laadusta	Ohjelma ei tee tarkistuksia käyttäjän antamille syötteille, esimeriksi numeroita sisältävissä kentissä voi olla kirjaimia.

3.2.3 Muutettavan datan löytäminen ja valitseminen

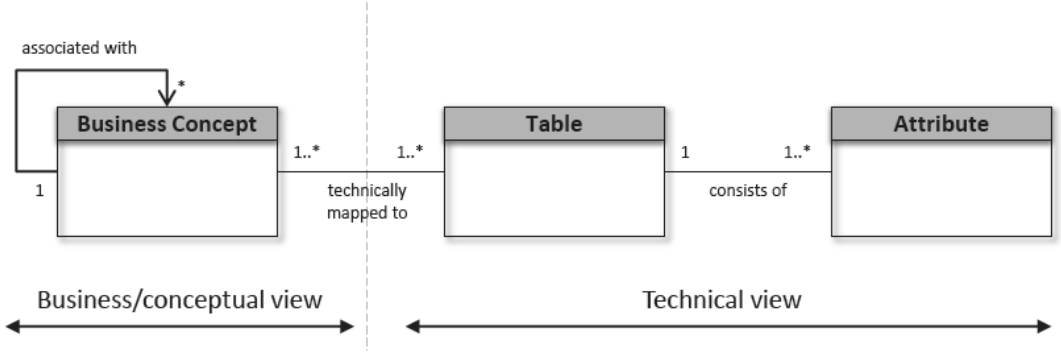
Muutettavan datan valitseminen tarkoittaa lähdejärjestelmän datan karsimista siten, että uuteen järjestelmään saadaan sen käytön kannalta kaikki tarpeellinen data. Muutettavan datan valitseminen ei pelkästään riitä, vaan muutettava data on myös löydettävä ja dokumentoitava. Datan valitsemisen/löytämisen apuna voidaan käyttää F. Matthesin [3 ss. 20-22] esittämää liiketoimintakonsepti-käsitettä.

Liiketoimintakonsepti (business concept) *on esitys loogisesta tai fyysisestä konseptista liiketoimintadomainissa, johon liittyy vähintään sen nimi, määritelmä, yhteydet, säännöt, käytännöt ja rajoitteet.* [3 s. 20].

Kuvassa 3.2 on esitettynä liiketoimintakonseptin ja tietokantarelaation väliset yhteydet UML-notaatiolla. Liiketoimintakonseptia voidaan ajatella tietokantarelaation abstraktiotasoa ylempänä abstraktiona, jolla on tietokantarelaatioon verrattuna seuraavia ominaisuuksia:

- Tietokantataulun attribuutit kuvaavat liiketoimintakonseptin attribuutteja teknisellä tasolla.
- Tietokantataulu ei välttämättä kuvaa koko liiketoimintakonseptia.
- Liiketoimintakonseptin attribuuteilla ei ole tietotyyppiä, kuten tietokantatauluilla.

Edellä mainituista ominaisuuksista seuraa ongelmia tilanteissa, joissa on tarkoitus muuttaa kahdesta samalla liiketoiminta-alueella toimivasta järjestelmästä dataa yhteen järjestelmään. Kahdessa eri järjestelmässä voidaan siis kuvata sama liiketoimintakonsepti teknisesti (tietokantarelaationa) eri tavalla, jolloin näiden kahden teknisen kuvauksen yhdistäminen voi olla ongelmallista.

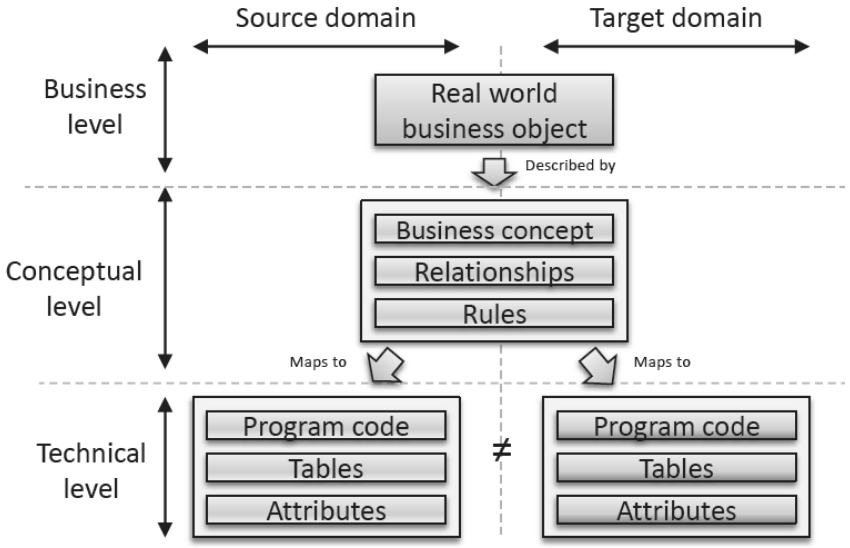


Kuva 3.2. Liiketoimintakonseptien ja relaatioiden välinen suhde. [3 s. 21]

Liiketoimintakonseptit voidaan pääasiassa tunnistaa kolmella eri tavalla. Ensimmäinen tapa on tutkia asiakkaan liiketoimintaprosesseja käyttäen apuna asiakkaan liiketoimintaa tuntevia henkilöitä. Toinen tapa on tutkia lähdesovelluksen dataa ja tunnistaa sieltä oleelliset liiketoimintakonseptit. Kolmas tapa on tutkia lähdesovelluksen työkulkuja lähdesovelluksen asiantuntijoiden avustuksella.

Liiketoimintakonsepteja voidaan hyödyntää tekemällä niistä esimerkiksi graafisia malleja, joiden avulla voidaan kommunikoida todellisen maailman liiketoimintakonsepteista yhtenäisellä terminologialla. Lisäksi liiketoimintakonsepteilla voidaan validoida

tuliko lähdejärjestelmästä löydetty tarpeellinen liiketoimintakonsepti esitettyä kohdejärjestelmässä (Kuva 3.3). Liiketoimintakonseptimallit auttavat erityisesti monimutkaisissa migraatioissa, joissa uuteen järjestelmään täytyy sisällyttää dataa jopa useista sadoista tauluista. [3 ss. 20-22]



Kuva 3.3. Liiketoimintakonseptit ovat samoja lähde domainissa ja kohde domainissa. [3 s. 22]

Liiketoimintakonsepti on yksi tapa mallintaa muutettavaa dataa liiketoimintatasolla. Käytännössä voidaan kuitenkin käyttää mitä tahansa työkalua, jolla liiketoiminnan käsitteet ja niiden väliset yhteydet saadaan mallinnettua. Tärkeintä on kuitenkin luoda malli, jonka avulla voidaan keskustella lähdejärjestelmän datan merkityksestä ja siitä, kuinka kyseinen data on tarkoitus mallintaa kohdejärjestelmässä. [1 ss. 41-45]

3.2.4 Datan valitsemisen lähestymistavat

Kirjallisuudessa yleisesti erotetaan kaksi lähestymistapaa datan valitsemiseen. Lähestymistavat ovat *source push* ja *target pull*.

Source push -lähestymistapa tarkoittaa sitä, että kaikki lähdejärjestelmän data viedään kohdejärjestelmään. Tämä idea voi kuulostaa aluksi hyvältä, sillä näinhän kohdejärjestelmään viedään varmasti kaikki oleellinen tieto. Tällöin tullaan kuitenkin vieneeksi paljon turhaa dataa, sillä lähdejärjestelmässä on paljon vanhaa, väärää tai tarpeetonta tietoa. K. Haller [7] mainitsee, että joissain tapauksissa vain 10 % attribuuteista täytyy muuttaa uuteen järjestelmään.

Target pull -lähestymistavassa kohdejärjestelmään "vedetään" ainoastaan sen tarvitsemat attribuutit. Näin vältytään siirtämästä turhaa tietoa, joka vain hidastaisi migraatiota ja mahdollisesti myös häiritsisi uuden järjestelmän toimintaa.

3.2.5 Uuteen järjestelmään siirtyminen

Kirjallisuudessa esitetään kolme eri tapaa siirtyä vanhasta järjestelmästä uuteen järjestelmään. Nämä tavat ovat big bang, inkrementaalinen ja rinnakkainen migraatio, jotka seuraavaksi esitellään.

Big bang -migraatiossa kaikki data viedään kohdesovellukseen kerralla. Tässä ongelmaksi voi muodostua muutettavan datan määrä, jolloin seuraa mahdollisia suorituskykyongelmia. Suorituskykyongelmia vastaan voidaan tietenkin taistella hankkimalla lisää laitekapasiteettia migraation suorittamiseen, olettaen tietysti migraatioprosessin skaalantuvan horisontaalisesti. Aina laitekapasiteetin lisääminenkään ei riitä tai se ei ole budjettisyistä mahdollista. Tällöin oleellista datamigraation onnistumiselle on riittävän pitkä aikaikkuna, jolloin tuotantojärjestelmä on pois käytöstä. Tästä saattaa muodostua ylitsepääsemätön este, mikäli asiakkaan liiketoiminnan kannalta pitkä käyttökatko (esimerkiksi 24h) ei ole mahdollinen. Taulukossa 3.2 esitetään big bang -migraation edut ja haitat. [3] [10] [1 s. 104]

Taulukko 3.2. Big bang migraation edut ja haitat.

Edut	Haitat
Vanhaa ja uutta järjestelmää ei tarvitse käyttää rinnakkain, siten säästetään kustannuksissa.	Kaikki tai ei mitään -asetelma
Nopein toteuttaa.	Mikäli ongelmia huomataan vasta migraation jälkeen, on vaikeaa palata vanhaan järjestelmän.
	Vaatii tuotantojärjestelmän alasajon pitkäksi ajaksi, mikäli lähdedataa on paljon.

Inkrementaalinen migraatio suoritetaan inkrementaalisesti, jolloin uuteen järjestelmään viedään tietty osa kokonaisdatasta kerralla. Tällöin haasteena on löytää datasta sopivat osiot, jotka voidaan viedä erikseen uuteen järjestelmään. Inkrementaalinen migraatio suoritetaan siten, että lähdejärjestelmän vastuulla olevia palveluita siirtyy vähitellen kohdejärjestelmään. Yhden palvelun siirtäminen kohdejärjestelmään ei saa kuitenkaan häiritä muiden palveluiden toimimista. Uuteen järjestelmään voidaan viedä esimerkiksi tiettyyn maahan tai toimipisteeseen liittyvä data kerrallaan. Taulukossa 3.3 esitetään inkrementaalisen migraation edut ja haitat. [3] [10] [1 s. 104]

Taulukko 3.3. Inkrementaalisen migraation edut ja haitat.

Edut	Haitat
Pienemmän datamäärän rollback on helpompaa.	Kahden rinnakkaisen tuotantojärjestelmän ylläpito vaatii lisätyötä. Sopivien dataosioiden tunnistaminen voi olla vaikeaa tai niitä ei ole olemassa.

Rinnakkaisessa migraatiossa data synkronoidaan lähde- ja kohdejärjestelmän välillä siten, että kumpaakin järjestelmää voidaan käyttää tuotannossa yhtä aikaa. Tämä lähestymistapa on kallein toteuttaa, mutta samalla riskit liiketoiminnalle ovat kaikista pienimmät. Rinnakkaisessa migraatiossa vanhaan järjestelmään voidaan palata takaisin milloin vain. Lisäksi voidaan validoida uuden järjestelmän tarkoituksenmukaisuus ja toimivuus tuotantokäytössä ja ottaa vanha järjestelmä pois käytöstä vasta sitten, kun ollaan varmoja uuden järjestelmän kelvollisuudesta. Taulukossa 3.4 esitetään rinnakkaisen migraation edut ja haitat. [10] [1 s. 104]

Taulukko 3.4. Rinnakkaisen migraation edut ja haitat.

Edut	Haitat
Riskit liiketoiminnalle ovat pienet, sillä vanhaan järjestelmään päästään helposti takaisin, mikäli uusi järjestelmä osoittautuu kelvottomaksi.	Kallis toteuttaa, sillä järjestelmien synkronoinnin toteuttaminen vaatii paljon ylimääräistä toteutus- ja testaustyötä. Hitain vaihtoehto sen monimutkaisuuden vuoksi.

3.3 Datamigraatioalustan pystyttäminen

Datamigraatioalustan pystyttämisessä (platform setup) käytännössä luodaan infrastruktuuri datamigraatiosovellusten ja tietovarastojen tueksi. Datamigraatioalustalla ajettavat sovellukset ja tietovarastot esiteltiin kohdassa 2.4, joten esitys sivutetaan tässä kohdassa.

3.4 Lähdedatan purku

Datamigraatioalustan pystytyksen jälkeen seuraa lähdedatan purkuvaihe, jossa data puretaan lähdejärjestelmästä välikerääntymisalueelle. Lähdedatan purku (source data unloading) -vaihe voi olla hyvin suoraviivainen tai erittäin työläs riippuen lähdesovelluksen tietovaraston dokumentaatiosta ja liitettävyydestä (esim. ODBC yhteensopivuus). Esi-

merkiksi huonossa tapauksessa tietokannan dokumentaatiota ei ole olemassa tai saatavilla, jolloin lähdesovelluksen tietokanta joudutaan käänteismallintamaan. Helpossa tapauksessa taulujen hakeminen lähdejärjestelmästä tarkoittaa vain triviaalien INSERT INTO -lauseiden kirjoittamista.

Datan purkamisen yhteydessä on hyvä suorittaa validointi puretulle datalle. Tämän validoinnin tarkoituksena on selvittää, saadaanko kaikki data ulos järjestelmästä ehjänä. Testaaminen on erityisesti tarpeen, mikäli datan purkamiseksi on jouduttu tekemään monimutkaisia ja siten virhealttiita työkaluja.

Tämän vaiheen ensimmäisen iteraation lopputuloksena lähdesovelluksen datat on tuotu lähdekerääntymisalueelle sellaiseen muotoon, jotta niitä voidaan käyttää datan transformaatiotyökalujen kehitystä varten. [3]

3.5 Lähdedatan analyysi

Lähdedatan analyysivaiheessa (source data analysis) työstetään katsaus lähdedatan laadusta ja rakenteesta. Datan laadun ja rakenteen analyysin tarkoitus on tunnistaa epäjohdonmukaisuudet, redundanssit, epätarkkuudet ja viitteiden eheyksien ongelmat, jotta myöhemmissä datan puhdistusvaiheissa osattaisiin ottaa ne huomioon. Lyhyesti datan laatu -käsite kuvaa kuinka hyvin data vastaa sen kuvaamaa todellisen maailman käsitettä.

Datan laatuongelmia ovat puutteelliset tai väärät tiedot. Esimerkiksi henkilötietoja käsittelevässä järjestelmässä saattaa henkilöltä puuttua henkilötunnus, jolloin kyseisen henkilön tunnistaminen yksiselitteisesti on vaikeaa.

Datan rakenteen analyysin tarkoitus on tunnistaa ongelmia tietokannan viitteiden eheydessä tai puutteellisia pääavaimia. Mikäli pääavaimet ovat viallisia tai niitä ei ole, joudutaan tunnistamaan pääavainkandidaatit analysoimalla dataa. Lisäksi analysoitavasta datasta voidaan kerätä statistista tietoa, kuten null-arvojen osuus tietyllä sarakkeella.

Datan laatusäännöt ovat metriikoiden määrittelyjä, joiden avulla analyysin kohteena olevan datan laatu mitataan. Datan laatusäännöt voivat mitata datan sisäisiä (esimerkiksi lukujen arvoalueella pysymistä) tai ulkoisia (esimerkiksi ovatko viitteet muihin tauluihin ehjiä) laatuongelmia. Yleisesti datan laatusääntöjen tarkoitus on mitata todellisen maailman ilmiöiden ja sitä mallintavan datan vastaavuutta. [1 ss. 59-73]

Esimerkiksi taulukossa 3.5 esitettyä jaottelua voidaan käyttää datan laadun luokitteluun tietovarastoittain. Jaottelussa tunnisteet A-D kuvaavat datan lähteen luotettavuutta ja tunnisteet 1-4 luodaan datan laatusääntöjen perusteella. [1 ss. 48-49]:

Taulukko 3.5. Datan laadun tunnisteet.

Tunniste	Merkitys
A	Data on täysin auditoitua tai peräisin reaaliaikajärjestelmistä.
B	Data on kerätty statistisesti oikealla osittais-auditoinnilla tai luotettavasti transformoitu luokan A järjestelmistä.

C	Data on kerätty sekundäärisistä järjestelmistä tai yhteen vedetty luokan B järjestelmistä.
D	Impressionististä dataa, joka on kerätty rajoitetulla selvityksellä tai on generoitu luokan C järjestelmistä.
1	Data on 98% tarkkaa mitattuna datan laatusääntöjen perusteella.
2	Data on 95% tarkkaa mitattuna datan laatusääntöjen perusteella.
3	Data on 85% tarkkaa mitattuna datan laatusääntöjen perusteella.
4	Data on vähemmän kuin 85% tarkkaa mitattuna datan laatusääntöjen perusteella.

Näiden tunnisteiden yhdistelmästä voidaan luoda tunniste tietovarastolle. Tietyt yhdistelmät kuten A4 ovat kuitenkin käytännössä mahdottomia, sillä täysin auditoitu järjestelmä tuskin tuottaa vähemmän kuin 85% tarkkaa dataa.

Tämä triviaalille kuulostava vaihe on kriittinen datamigraatioprojektin onnistumisen kannalta, sillä laadultaan heikko data voi kohdejärjestelmään päästessään aiheuttaa ongelmia tai vaikeuttaa datamigraation onnistumista.

Datan laadun analysointia varten voidaan kehittää omia työkaluja tai käyttää markkinoilta saatavia työkaluja. Esimerkiksi Informatica [11] myy liiketoimintadomainista riippumatonta datan laadun analyysiin tarkoitettua työkalua. [3]

3.6 Lähdedatan puhdistaminen

Lähdedatan puhdistamisessa (source data cleansing) korjataan edellisessä vaiheessa havaittuja virheitä. Datan puhdistus on tärkeää, koska kohdejärjestelmällä saattaa olla tiukemmat säännöt datan oikeellisuutta koskien. Tällöin virheellistä dataa ei voida yksinkertaisesti viedä uuteen järjestelmään. Lisäksi datan puhdistamiseksi luotuja puhdistamissääntöjä voidaan hyödyntää jatkossa yrityksen datan laadun parantamiseen. [1 s. 18]

Datan puhdistus voidaan suorittaa rinnakkaisena tehtävänä muun datamigraatioprosessin kanssa, jolloin sitä suorittamaan voidaan osoittaa omat resurssit. Mikäli näin tehdään, datan puhdistaminen ei pidennä datamigraatioprojektin aikataulua kalenteriajassa.

Datan puhdistamista voidaan suorittaa useissa eri datamigraatioprosessin vaiheissa datan siirtämisen aikana tai paikallaan tietovarastoissa. Kullakin vaiheella on etunsa ja haittansa, joten datan puhdistamisen ajankohdan valitseminen vaikuttaa lopputulokseen. [3 ss. 39-41] [12]

Data voidaan puhdistaa jo **lähdetietokannassa** (Taulukko 3.6), ennen kuin sille tehdään mitään muita operaatioita. Tällöin datan puhdistamiseen tarvittavien työkalujen ja prosessien kehitys voidaan aloittaa jo kauan ennen datamigraatioprojektin aloittamista.

Taulukko 3.6. Datan puhdistaminen lähdetietokannassa.

Edut	Haitat
Datan puhdistaminen voidaan aloittaa ennen kuin datamigraatioprojekti aloitetaan. Mikäli virheellisen datan puhdistus tehdään etukäteen, se ei pääse aiheuttamaan ongelmia datamigraatioprosessin muissa vaiheissa. Käytettävissä on vanhaa ohjelmistoa käyttävien ihmisten asiantuntemus, he voivat jo valmiiksi tietää kuinka virheellinen data korjataan.	Vanha järjestelmä on käytössä, jolloin sinne jatkuvasti syntyy uutta virheellistä dataa. Tämä monimutkaistaa puhdistusprosessia.

Data voidaan puhdistaa **kohdetietokannassa** käyttöönoton jälkeen. Tämä vaihtoehto on heikko ja siihen tulisi turvautua vain jos datan laatuun liittyviä ongelmia ei ole tunnistettu ennen käyttöönottoa (Taulukko 3.7).

Taulukko 3.7. Datan puhdistaminen kohdetietokannassa.

Edut	Haitat
Säästää resursseja datamigraatioprosessin rakennusvaiheessa.	Viallinen data voi aiheuttaa ongelmia kohdejärjestelmässä ja sen kautta uuden järjestelmän luotettavuus voi kärsiä käyttäjien silmissä. Viallinen data saattaa jäädä myös kokonaan puhdistamatta, koska tässä vaiheessa uuden järjestelmän käyttöönottoprojektia huomio ei ole enää datamigraatiossa. Tämä ongelma realisoituu erityisesti, jos datamigraatioprojekti on ylittänyt budjetin tai aikataulun.

Osa datan puhdistamisesta voidaan suorittaa myös tiedon purkamisen, transformaation tai viemisen aikana. Nämä työvaiheet kuitenkin tehdään tuotannollisessa migraatiossa käyttökatkon aikana. Tällöin on huomioitava, ettei aikaa välttämättä ole mahdollisille manuaalisille korjauksille.

3.7 Datan transformaatio

Datan transformaatiossa (data transformation) määritellään datan transformaatiosäännöt sekä liiketoiminta, että teknisellä tasolla. Tämän jälkeen tuotetaan datan tekniset transformaatiosäännöt, jotka määritellään ohjelmakoodina. Transformaatiosäännöistä puolestaan koostetaan transformaatio-ohjelma, jossa transformaatiosäännöt toimivat aliohjelmina.

Datan transformaatiovaihe on iteratiivistä ohjelmistokehitystä (Kuva 3.4), missä kehitetään ohjelmia (transformaatiosääntöjä) määrittelyiden (liiketoimintatransformaatiosäännöt) perusteella. Tämän vaiheen lopputuloksena on tuotettu dataa kohdekeräämismisalueelle, joka on valmiina vietäväksi uuteen järjestelmään. [3 ss. 41-46]

Kuva 3.4. Teknisten transformaatiosääntöjen suunnitteluprosessi.

3.7.1 Transformaatiosäännöt

Liiketoimintatason transformaatiosääntöjen on tarkoitus antaa riittävä määrittely teknisten transformaatiosääntöjen laatimiselle. Nämä määrittelyt voivat olla vähemmän formaaleja. Esimerkiksi asiakkaan tietojärjestelmän asiantuntija voi esittää tekstuaalisen määritelmän, kuinka tieto esitetään vanhassa järjestelmässä ja kuinka se on tarkoitus esittää uudessa.

Tekniset transformaatiosäännöt puolestaan ilmaistaan aliohjelmina, joista kukin esittävät yhden lähde-attribuutin kuvausta kohde-attribuutiksi. Transformaatio-sääntöjä voidaan kirjoittaa kaikilla ohjelmointikielillä, mutta usein on tarpeen ottaa huomioon tehokkuuskysymykset datan määrästä riippuen. Taulukossa 3.8 on esitettynä eri teknisten transformaatiosääntöjen tyyppejä. [3 s. 44]

Taulukko 3.8. Teknisten transformaatiosääntöjen eri tyyppejä.

Säännön nimi	Kuvaus
Suora kuvaus (Direct mapping)	Lähde-attribuutin arvo kuvataan 1:1 yhteen kohdeattribuuttiin.
Vakio arvo (Constant value)	Kohde-attribuutti täytetään jollain vakiolla (esim "1").
Oletus arvo (Default value)	Kohde-attribuutti täytetään vakio arvolla. Sääntö on voimassa vain jos kartoitusta kohdeattribuutille ei ole määritelty.
Toimialue kuvaus	Lähdetoimialueen arvot kuvataan suoraan kohdetoimialueen

(Domain mapping)	arvoihin (esim. "$" - "USD" ja "€" - "EUR").
Kuvaustaulu (Mapping table)	Kohdeattribuutin arvo haetaan kuvaustaulusta lähdeattribuutin arvon perusteella. (esim. maan nimi - maan koodi)
Kuvausfunktio (Mapping function)	Yksi tai useampi lähdeattribuutti kartoitetaan kohdeattribuutiksi kuvausfunktion avulla. (esim. lämpötilakonversio Celciuksesta Farenheitiksi.)
Luotu arvo (Generated value)	Kohdeattribuutin arvo luodaan riippumatta lähdedatasta. (esim. asetetaan nykyinen päivämäärä).

3.7.2 Proseduraaliset vs deklaratiiviset lauseet

Tietokantamoottorit tarjoavat hyvät työkalut käsitellä tehokkaasti suuria määriä dataa, mutta proseduraalisella ajattelulla voidaan kompastua SQL-kyselyiden kirjoittamisessa. SQL-lauseet kannattaa tehokkuutta ajatellen kirjoittaa deklaratiivisiksi lauseiksi, sillä tietokantamoottoreilla on mahdollisuus optimoida niitä. [13]

Ohjelmassa 1 on esitettynä proseduraalinen esimerkki varastossa olevien tarvikkeiden määrän selvittämisestä käyttäen SQL-kursoria.

```
DBCC FREEPROCCACHE;
GO

DECLARE @StartTime datetime; SET @StartTime = GETDATE();
DECLARE @CurrentItems bigint; SET @CurrentItems = 0;
DECLARE @TranCode char(3), @Items int;
DECLARE MyCursor CURSOR
    FOR SELECT TranCode, Items FROM Stock

OPEN MyCursor
FETCH NEXT FROM MyCursor INTO @TranCode, @Items
WHILE @@FETCH_STATUS = 0
BEGIN
    IF @TranCode IN ('IN', 'RET')
         SET @CurrentItems = @CurrentItems + @Items;
    ELSE
         SET @CurrentItems = @CurrentItems - @Items;
    FETCH NEXT FROM MyCursor INTO @TranCode, @Items
END
CLOSE MyCursor
DEALLOCATE MyCursor

SELECT 'cursor', DATEDIFF(ms, @StartTime, GETDATE()), @CurrentItems

GO
```

Ohjelma 1. Varastossa olevien tarvikkeiden määrän selvittäminen käyttäen kursoria. [14]

Ohjelmassa 2 vastaava operaatio on tehty deklaratiivisesti eksplisiittisesti iteroimatta tulosjoukon läpi.

```
DBCC FREEPROCCACHE;
GO
USE testdb;
DECLARE @StartTime datetime; SET @StartTime = GETDATE();
DECLARE @CurrentItems bigint; SET @CurrentItems = 0;

SELECT @CurrentItems = @CurrentItems +
    CASE TranCode
         WHEN 'IN' THEN Items
         WHEN 'RET' THEN Items
         WHEN 'OUT' THEN -Items
    END
FROM Stock

SELECT 'set', DATEDIFF(ms, @StartTime, GETDATE()), @CurrentItems
GO
```

Ohjelma 2. Varastossa olevien tarvikkeiden määrän selvittäminen deklaratiivisesti käyttäen SELECT-lausetta. [14]

Ohjelman 1 suoritusaika miljoonalle riville on Microsoft SQL Server 2008 R2-tietokannalla 18900ms ja ohjelman 2 490ms, jolloin ohjelman 2 suoritusaika on vain 2,6 % ohjelman 1 suoritusajasta. Imperatiivisella SQL-ohjelmoinnilla voidaan siis merkittävästi hidastaa koko datamigraatioprosessia.

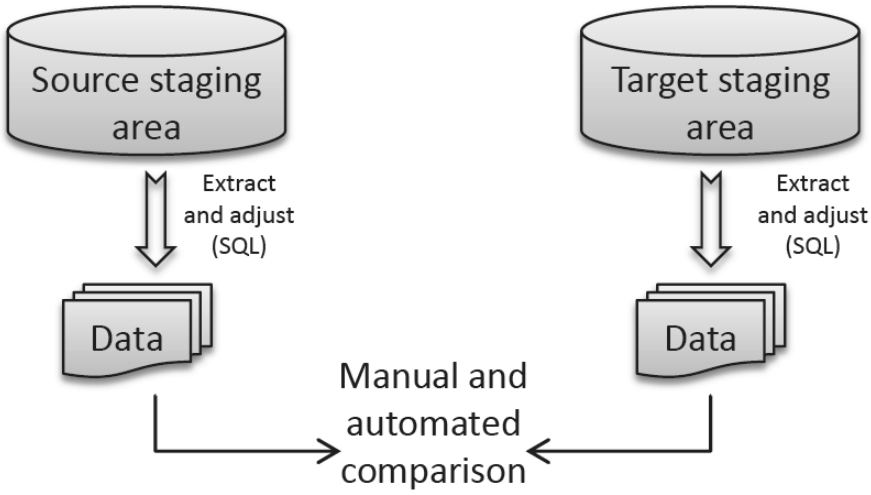
3.8 Datan validaatio

Datan transformaatio-ohjelman kehittämisen ja suorittamisen jälkeen kohdekerääntymisalueella on dataa, joka täytyy validoida. Datan validoinnin tarkoitus on varmistaa uuteen järjestelmään vietävän datan täydellisyys, oikeellisuus ja johdonmukaisuus. Suurin haaste tässä vaiheessa on lähdedatan ja kohdedatan vertaaminen, sillä kohdetietorakenne poikkeaa usein oleellisesti lähdetietorakenteesta. Lisäksi dataa on mahdollisesti kuvattu uudelleen. [3 s. 47]

Kohdedataa verrataan lähdedataan käyttämällä yhdistelmää seuraavista vaihtoehdoista:

- **Automaattinen vertailu** varmistaa oikeellisuuden, täydellisyyden ja yhdenmukaisuuden rakenteellisella ja data-tasolla. Automaattisella vertailulla saadaan aikaan hyviä tuloksia nopeasti, mutta automaattisten testien implementointi vie aikaa.
- **Manuaalinen vertailu** tarkoittaa datan validointia käyttäen apuna datan analyysityökaluja. Manuaalisella vertailulla voidaan suorittaa esimerkiksi pistokokeita.

Jotta lähde ja kohdedata olisivat vertailukelpoista, täytyy dataa muokata (Kuva 3.5). Tässä vaiheessa on hyvä ottaa huomioon, että ollaan validoimassa transformaatio-ohjelman oikeellisuutta, jolloin ei kannata uudelleenkäyttää mahdollisesti väärää transformaatio-ohjelman koodia. Tällöin pätee myös ohjelmistotestauksessa tunnettu käytäntö: ohjelman kehittäjän tulee olla eri henkilö kuin ohjelman testaajan, sillä muuten samat virheet pääsevät helposti toistumaan testikoodia kehitettäessä. [3 ss. 47-49]



Kuva 3.5. Lähde ja kohdedatan vertailu. [3 s. 48]

Tämän vaiheen lopputuloksena kohdekerääntymisalueella oleva data on huolellisesti testattu.

3.9 Datamigraatioprosessi-testit

Datamigraatioprosessi-testit (data migration process tests) toimivat kuivaharjoitteluna koko datamigraatioprosessille. Testin tarkoituksena on osoittaa, että koko datamigraatioprosessi saadaan vietyä alusta loppuun riittävän nopealla aikataululla, jotta se on mahdollista suorittaa tuotannossa datamigraatiolle avautuvan aikaikkunan sisällä. Datamigraatioprosessi koostuu ohjelmien lisäksi myös ihmisten toiminnasta, jonka merkitys datamigraatioprosessissa täytyy dokumentoida. Syntynyt dokumentti toimii ohjeistuksena datamigraatioprosessin suorittamiselle. Tämän vaiheen lopputuloksena on valmisteltu, testattu ja dokumentoitu datamigraatioprosessi. [3 s. 50]

3.10 Kohdesovelluksen testaus

Kohdesovelluksen testaus (target application tests) -vaiheen tarkoitus on varmistaa, että kohdesovellus toimii sinne viedyn datan kanssa. Kohdesovellukseen viedään kaikki data lähdejärjestelmästä jonka jälkeen sen toimivuutta testataan hyödyntäen käyttötapauksia. Lisäksi sovellus voidaan ottaa mahdollisuuksien mukaan pilottikäyttöön. Tällöin kohdesovelluksen lopulliset käyttäjät pääsevät ilmaisemaan mielipiteensä uuden järjestelmän toiminnasta.

Ongelmia kohdesovelluksen toiminnassa voi esiintyä seuraavista syistä:

1. Dataa ei viety oikein uuteen sovellukseen.
2. Kohdesovelluksessa on virheitä.
3. Kohdejärjestelmä ei pysty käsittelemään sinne viedyn datan määrään ja hidastuu käyttökelvottomaksi.

4. Käyttäjä käyttää uutta järjestelmää väärin, jolloin järjestelmän käytön ohjeistusta on parannettava.

Virheen sattuessa on tärkeää tunnistaa missä prosessin vaiheessa virhe sijaitsee. Tämä voi olla vaikeaa, sillä virheen alkuperä saattaa olla lähdesovelluksen virheellisessä datassa, migraatioprosessissa tai kohdesovelluksessa. Mikäli virhe on kohdesovelluksessa, jää vastuu virheen korjaamisesta kohdesovelluksesta vastuussa olevalle sidosryhmälle. Tästä voi koitua ongelmia migraatioprojektille, koska kohdesovelluksesta vastuussa oleva sidosryhmä saattaa työskennellä eri yrityksessä, jonka vastuualueena ei ole datamigraation suorittaminen. [3 s. 51]

3.11 Integraatiotestit ja lopullinen harjoitus

Integraatiotestit ja lopullinen harjoitus (integration tests and final rehearsal) ovat viimeiset askeleet ennen tuotannollista migraatiota. Vaiheen tarkoituksena on varmistaa kohdesovelluksen toimivuus ennen sen käyttöönottoa. Lisäksi datamigraatioprosessi testataan harjoittelemalla sen suorittaminen alusta loppuun.

Integraatiotestien tavoite on varmistaa kohdesovelluksen ja siihen liittyvien sovellusten yhteistoiminta niihin vietävän datan kanssa. Integraatiotestit suoritetaan noudattaen hyviä ohjelmistojen testauksen periaatteita.

Lopullinen harjoitus on viimeinen testi datamigraatioprosessille ennen tuotannossa ajettavaa migraatiota. Harjoituksessa ajetaan datamigraatioprosessi täydellisenä alusta loppuun, niin kuin se on tarkoitus ajaa tuotannollisessa migraatiossa. Jos harjoitus toistetaan useita kertoja, voidaan harjoitusten pohjalta tuottaa "datamigraatiokäsikirjoitus". Datamigraatiokäsikirjoitus toimii käsikirjoituksena tuotannolliselle migraatiolle. Käsikirjoitus sisältää yksityiskohtaiset kuvauksen migraatiovaiheiden suorittamiseksi peräkkäisesti. Datamigraatiokäsikirjoitus on siis rakenteeltaan kuin ruoanvalmistusohje, joka esimerkiksi kertoo mitä pitää keittää ja kuinka kauan se kestää.

Datamigraatiokäsikirjoituksen avulla itse tuotannolliseen migraatioon kuluva aika saadaan pienemmäksi, sillä migraatioita suorittavien henkilöiden ei tarvitse miettiä "mitä teen seuraavaksi", vaan kyseiseen kysymykseen saa vastauksen suoraan datamigraatiokäsikirjoituksesta.

Datamigraatiokäsikirjoituksen lisäksi harjoitusten lopputuloksena saadaan arvokasta tietoa datamigraatioprosessin tehokkuudesta. Harjoitusten lopputulosten perusteella voidaan tehdä päätös datamigraation suorittamisesta, eli siirtymisestä tuotannolliseen migraatioon. [3 ss. 53-54]

3.12 Tuotannollinen migraatio ja viimeistely

Tuotannollinen migraatio ja viimeistely (productive migration and finalizing) ovat datamigraatioprojektin viimeiset vaiheet. Tuotannollisessa migraatiossa datamigraatioprosessi suoritetaan datamigraatiokäsikirjoituksen perusteella.

Tuotannollisen migraation aloittamiseksi täytyy ensin löytää sille sopiva aikaikkuna. Paras aikaikkuna migraatiolle on tietysti aikaväli jona järjestelmää ei käytetä (esimerkiksi yöllä), näin saadaan minimoitua datamigraatiosta liiketoiminnalle aiheutuva haitta. Tuotannollinen migraatio alkaa lähdejärjestelmän alasajolla, siten lähdedata on jäädytetty. Tämän jälkeen datamigraatio suoritetaan datamigraatiokäsikirjoituksen perusteella.

Datamigraation suorittamisen jälkeen kohdejärjestelmä otetaan tuotantokäyttöön, tätä hetkeä kutsutaan "ei paluuta -pisteeksi" (point of no return). Ei paluuta -pisteen jälkeen vanhaan järjestelmään paluu on erittäin vaikeaa, ellei siirtymisstrategia sisällä tietojen synkronointia vanhan ja uuden järjestelmän välillä. Tällöin vanhan järjestelmän data pysyy jatkuvasti ajan tasalla uuden kanssa, jolloin vanhaan järjestelmään voidaan palata helposti takaisin.

Viimeistely tarkoittaa yksinkertaisesti projektin päättymisen toteamista muodollisesti. Asiakkaan edustajat ja datamigraation toimittava osapuoli sopivat yhdessä, että projekti on toimitettu. Tässä vaiheessa vastuu järjestelmän toiminnasta siirtyy yleensä asiakkaan IT-hallinnon vastuulle. [3 ss. 55-56]

3.13 Projektinhallinta

Datamigraatioprojekteissa täytyy tasapainotella aikataulun, laadun ja kustannusten välillä, kuten muissakin IT-projekteissa. Datamigraation projektinhallinnalla on muutamia erityisominaisuuksia ohjelmistoprojektin hallintaan verraten, näitä erityisominaisuuksia esitellään tässä kohdassa.

Projektin sisäiset rinnakkaisuudet tarkoittavat, että datamigraatioprojekteissa tarvitsee usein hallita kolmea eri rinnakkaista toisistaan riippuvaa työnkulkua:

1. Lähdesovelluksen ylläpito
2. Kohdesovelluksen kehittäminen tai konfigurointi
3. Datamigraation suorittaminen

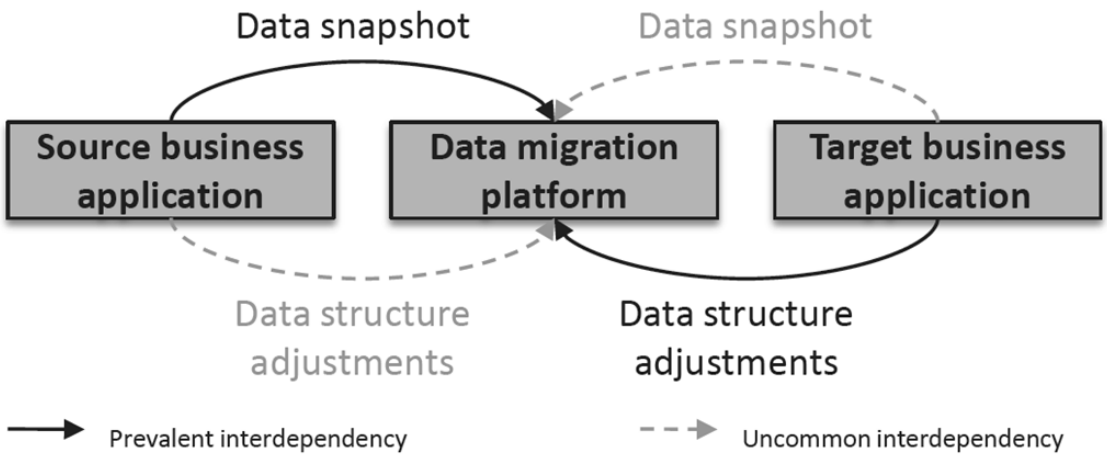
Näillä työnkuluilla esiintyy kuvan 3.6 mukaisia yleisiä riippuvuuksia (prevalent interdependency) sekä ei niin yleisiä, mutta mahdollisia riippuvuuksia (uncommon interdependency). Lähdedatasta joudutaan ottamaan uusia näytteitä silloin tällöin, jotta datamigraatio-ohjelmien toimivuutta voidaan testata tuoreella lähdedatalla. Joskus lähdesovelluksen tietorakenne voi muuttua hieman projektin aikana, varsinkin jos datamigraatioprojekti kestää pitkään. Jos kohdesovellusta kehitetään tai konfiguroidaan on hyvin yleistä, että kohdesovelluksen tietorakenteeseen tulee muutoksia. Lisäksi kohdesovelluksessa voi olla jo dataa (esimerkiksi järjestelmien yhdistämisessä), jolloin kohdejärjestelmän datasta joudutaan ottamaan näytteitä datamigraatioalustaan.

Lisäksi datamigraatio on usein nopeampi vaihe suorittaa, kuin uuden järjestelmän kehittäminen tai konfigurointi. Tällöin datamigraation suorittaminen voi joutua odottamaan uuden järjestelmän valmistumista.

Datamigraatioprojektin toimittamisen suorituskyky on yleensä paras kun vain yksi yritys toimittaa projektia. Tällöin vastuu projektista on yhdellä yrityksellä eikä vas-

tuuta tarvitse jakaa usealle yritykselle. Jos datamigraatioprojektia toimittaa useampi yritys, niin on lähes mahdotonta esittää yksityiskohtaisesti ja kirjallisesti kaikkia yksittäisen yrityksen velvollisuuksia datamigraatioprojektissa.

Kommunikaation määrä vaihtelee datamigraatioprojektin kuluessa datamigraation toimittajan, kohdesovelluksen toimittajan ja asiakkaan sidosryhmien edustajien välillä datamigraatioprosessimallin eri vaiheiden aikana. Esimerkiksi strategiavaiheen aikana täytyy kerätä paljon tietoa lähde- ja kohdesovelluksista sekä asiakkaan liiketoiminnasta ja sen datamigraatiolle asettamista rajoitteista. Transformaatiovaiheen aikana kommunikaation tarve vähenee ja transformaatiosovelluksia kehitetään kerättyjen tietojen perusteella. Testauksen aikana datamigraatioprosessin osiin täytyy tehdä viimeiset korjaukset, joten kommunikaation määrä myös sitä myöten kasvaa. [3 ss. 57-59]



Kuva 3.6. Lähde- ja kohdesovellusten sekä datamigraatioalustan väliset riippuvuudet. [3 s. 57]

4 ATOSTEKIN TOIMITTAMA DATAMIGRAATIO-PROJEKTI

Tässä luvussa kuvataan Atostekin toimittama osa datamigraatioprojektia. Salassapitovelvollisuuden vuoksi todelliset projektiin osallistuneet yritykset ja asiakkaan liiketoiminta-alue on nimetty uudelleen.

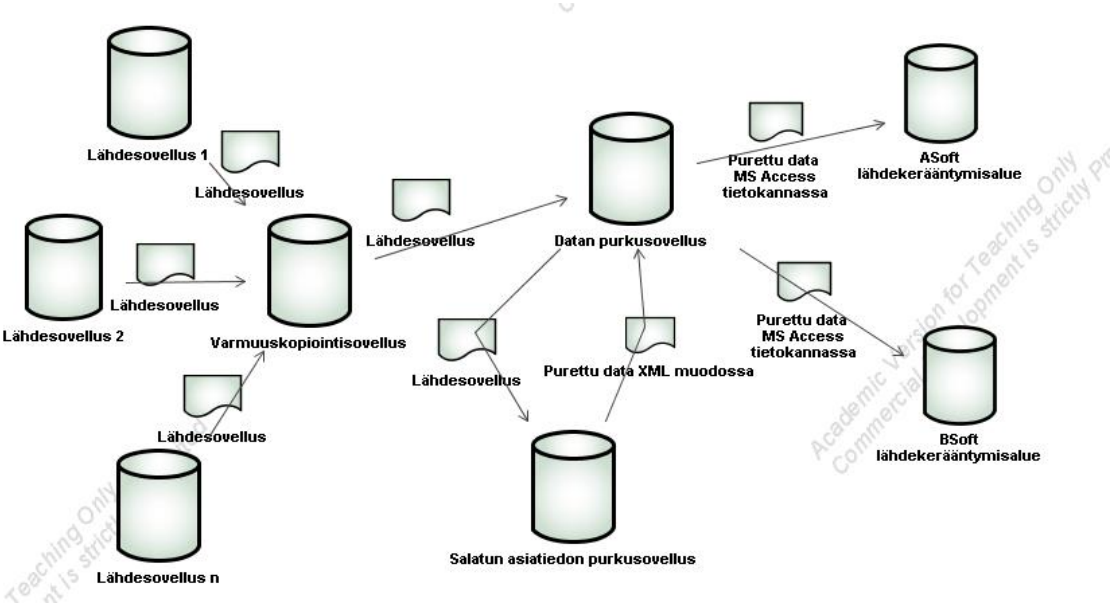
4.1 Yleiskuvaus

Tietojärjestelmäuudistuksen motivaationa asiakkaan näkökulmasta oli yhdistää omissa sisäverkoissa olevien ja valtakunnallisesti hajautettujen toimipisteiden tietokannat yhteen keskitettyyn tietokantaan. Tällöin eri toimipisteissä vierailevaa henkilöä voidaan palvella paremmin. Asiakkaan vanha DOS-pohjainen merkistökäyttöliittymällä varustettu järjestelmä päädyttiin korvaamaan kolmella web-selainkäyttöisellä järjestelmällä:

1. **Asianhallintajärjestelmä** asiakkaan liiketoimintasubstanssin hallitsemiseen.
2. **ERP-työkalu** asiakkaan resurssien hyödyntämisen suunnitteluun, lähinnä aikataulutukseen.
3. **Taloushallinnon järjestelmä** myynti- ja varastotietojen hallitsemiseen.

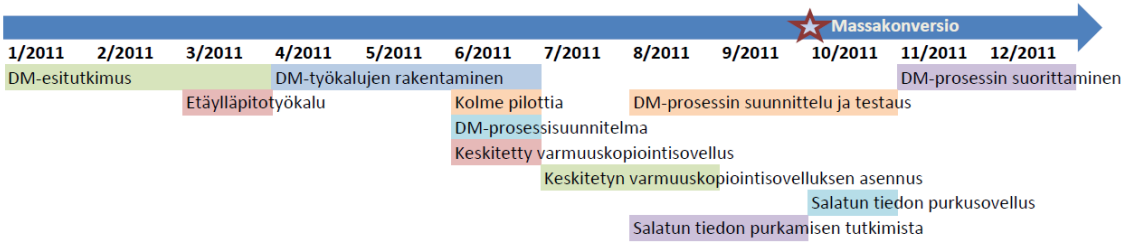
Atostekin toimittama osuus datamigraatiokokonaisuudessa keskittyi lähdesovelluksen tiedon purkamiseen. Tiedon purkamisen päämäärä oli saattaa tieto sellaiseen muotoon, että ASoft ja BSoft pystyivät suorittamaan datan transformaation kohdejärjestelmiin. Datan rakenteen määrittelynä toimi MS-Access tietokantaskeema, jonka taulut täytettiin lähdesovelluksen datalla ja toimitettiin ASoftille ja BSoftille. Migraation kohteena olevaa dataa oli noin 26GB tallennettuna SQL-tietokantoihin, jotka olivat hajautuneet ympäri Suomea asiakkaan toimipisteisiin.

Datan purku tapahtui asentamalla kaikkiin asiakkaan 70 toimipisteeseen keskitetysti hallittu varmuuskopiointisovellus, jonka avulla tiedot ladattiin datan purkusovellukseen. Datan purkusovellus koostui salaamattoman tiedon purkusovelluksesta ja salatun tiedon purkusovelluksesta. Datamigraatioon liittyvien sovellusten suoritusta hallittiin erillisellä orkestrointikomponentilla. Orkestrointikomponentti automatisoi osan datamigraatioprosessia. Datan purkamisen jälkeen data siirrettiin ASoftin ja BSoftin kohdekerääntymisalueille transformaatiota varten (Kuva 4.1).



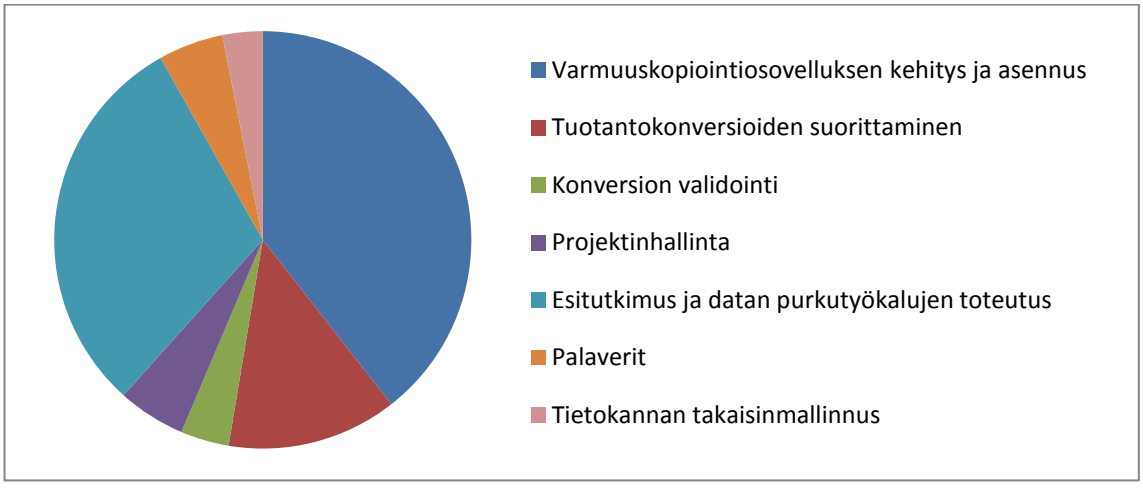
Kuva 4.1. Datan liikkuminen lähdesovelluksista lähdekerääntymisalueelle.

Tässä luvussa esitetään tarkemmin kuvassa 4.2 esitetyt datamigraatioprojektin vaiheet.



Kuva 4.2. Datamigraatioprojektin aikajana.

Projektin ajankäyttö jakaantui kuvan 4.3 mukaisesti. Vaiheet ovat peräisin Atostekin tuntien seurantajärjestelmästä, jossa ne ovat eri nimikkeillä verrattuna kuvan 4.2 vaiheisiin. Kuvasta on nähtävissä kuinka paljon työtä vaati datan nopean lataamisen mahdollistaminen asiakkaan toimipisteistä verrattuna datamigraatiotyökalujen tuottamiseen tai tuotantokonversioiden suorittamiseen.



Kuva 4.3. Projektin ajankäyttö.

4.2 Datamigraation esitutkimus

Esitutkimuksen aikana tehtiin toteuttamiskelpoisuus-tutkielma, jolla varmistettiin että datamigraatio on mahdollista suorittaa. Tässä kohdassa kuvataan esitutkimusvaiheen lähtötilanne ja sen aikana esiin tulleet ongelmat.

4.2.1 Lähtötilanne

Datamigraation lähdesovelluksena toimi vanha Microsoft DOS-käyttöjärjestelmälle suunniteltu myynti-, varasto- ja asiatiedon hallintaan tarkoitettu sovellus. Sovelluksen tietokantamoottorina toimi transaktionaalinen, mutta ei relationaalinen Pervasiven Btrieve record manager. Sovelluksen tietokannan skeemoja ei ollut saatavilla, vaan ne jouduttiin takaisinmallintamaan päättelemällä taulujen nimet ja sarakkeet sovelluksen datan perusteella käyttäen Pervasiven DDF (Data Definition File) Builder-työkalua. Esitutkimuksen aikana huomattiin, että tiedot saadaan vietyä Btrievestä Pervasive SQL-tietokantaan hyödyntäen Pervasive Control Center-sovellusta.

4.2.2 Ongelmat

Esitutkimuksen myötä havaittiin seuraavat ongelmat:

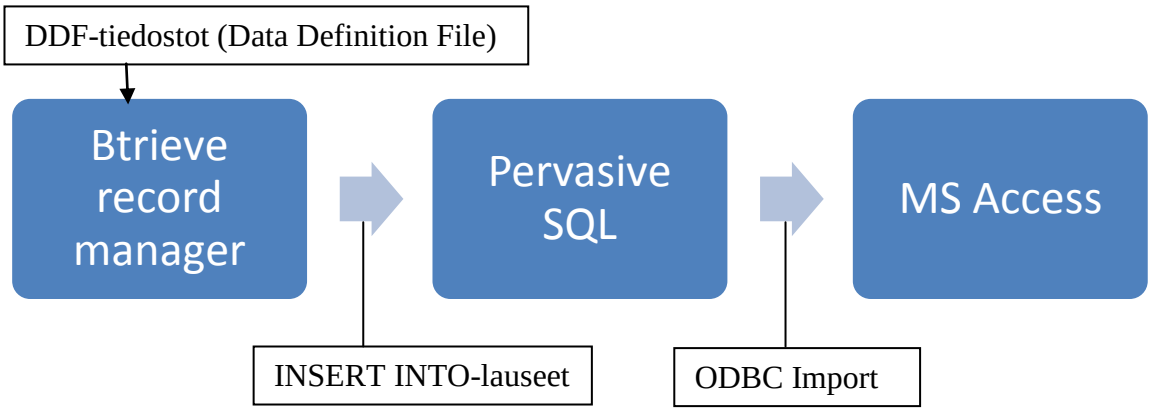
- Lähdesovelluksen tietokantana toimii vanha Pervasive Softwaren transaktionaalinen Btrieve-tietokantamoottori, jonka tietokantatiedostot eivät sisällä tietokantataulujen metatietoja.
- Tietokannan rakenne joudutaan takaisinmallintamaan sovelluksen dataa hyväksi käyttäen.
- Sovelluksen lähdekoodeja tai dokumentaatiota ei ole saatavilla.
- Asiakkaan toimipisteet ovat hajautuneet maantieteellisesti ympäri Suomea, jolloin vierailleminen kaikissa toimipisteissä on hyvin kallista.
- Btrieven ja Pervasive SQL:n tietotyypit eivät ole kaikki yhteensopivia vaan suuri osa joudutaan konvertoimaan.
- Toimipisteiden hajautuminen valtakunnan laajuisesti on myös ongelma.
- Toimipisteisiin on asennettuna eri versioita lähdesovelluksesta.
- Sovellusta on käytetty eri tavoilla eri toimipisteissä.

Esitutkimuksen jälkeen Atostek lähetti asiakkaalle tarjouksen datamigraation toimittamisesta.

4.3 Datamigraatiotyökalujen rakentaminen

Datamigraatiotyökalujen rakentaminen eteni kahdessa vaiheessa:. Ensin takaisinmallinnettiin tietokantataulujen metatiedot hyödyntäen Pervasiven DDF-builder työkalua, sitten kirjoitettiin datan konversion suorittavat funktiot ja INSERT INTO -lauseet. Datan konversiofunktioiden tehtävänä oli tulkita tyypittämätön data SQL-tietotyypeiksi.

Pervasive SQL:stä datat ladattiin MS-Access tietokantaan ODBC-rajapinnan avulla (Kuva 4.4). Tuloksena saadut MS-Access tietokantatiedostot toimitettiin eteenpäin ASoftille ja BSoftille datan transformaatiota varten.



Kuva 4.4. Datan konversion vaiheet.

Tässä vaiheessa myös valittiin kohdesovellukseen vietävä data. Asiakkaan kanssa pidettiin palaveri ja esiteltiin purettua Pervasive SQL -kantaa, jolloin asiakas pystyi kertomaan mitä dataa he tarvitsevat kohdesovelluksessa.

4.4 Etäylläpitotyökalu

Etäylläpitosovellus otettiin käyttöön, koska vierailleminen asiakkaan 70 toimipisteessä olisi ollut hyvin kallista. Laskelmien mukaan työaikaa olisi kulunut kutakin toimenpidettä varten noin vuorokausi per toimipiste. Etätukisovelluksen avulla etäyhteys toimipisteisiin saatiin puolestaan noin 30 minuutissa mukaan lukien sovelluksen asennukseen kuluvan ajan. Etätukisovellukselta vaadittiin seuraavat ominaisuudet:

- Sovelluksen täytyy päästä läpi palomuureista.
- Sovellus tukee tiedostojen siirtoa.
- Sovellus tarjoaa mahdollisuuden kontrolloida asiakkaan Windows-konetta etäyhteyden yli.
- Sovelluksen täytyy toimia vaikka molemmat päät ovat NAT:in takana.
- Sovelluksen asennuksen tulee onnistua myös ei tekniikkaan orientoituneelta henkilöltä puhelimitse opastettuna.

Etäylläpitosovellukseksi valittiin avoimen lähdekoodin ChunkVNC-sovellus [15]. ChunkVNC on etätukeen tarkoitettu UltraVNC [16] -kääre, joka koostuu kolmesta osasta:

InstantSupport: InstantSupport on kustomoitava AutoIT-skripti, joka käännetään suoritettavaksi tiedostoksi. Kun tämä tiedosto suoritetaan niin se purkaa UltraVNC-palvelimen väliaikaiseen hakemistoon, luo satunnaisen tunnisteen ja muodostaa salatun yhteyden Repeateriin. Tämä sovellus jaettiin asiakkaalle web-käyttöliittymän kautta.

Repeater: Repeater asennetaan julkisessa verkossa olevalle Atostekin palvelimelle. Sen tehtävänä on vastaanottaa InstantSupport-etätukipyyntö ja yhdistää kyseinen pyyntö Vieweriin.

Viewer: Viewer toimii VNC-asiakkaana, joka ottaa salatun yhteyden Repeaterin kautta InstantSupport-palvelimeen siltä saadun etätukitunnisteen avulla. Viewer toimii tämän jälkeen kuin muutkin VNC-ohjelmat.

Repeater asennettiin toimimaan portissa 443, joka on sama kuin salatun http-yhteyden käyttämä portti. Tämä siksi, että kyseinen portti on yleensä avattuna palomuureissa, joten näin saatiin ohitettua mahdolliset palomuurit ja etäyhteyden muodostaminen onnistui palomuureista huolimatta.

4.5 Keskitetty varmuuskopiointisovellus

Varmuuskopiointisovelluksella oli tärkeä tehtävä datamigraatioprosessissa. Sen avulla saatiin nopeasti viimeisimmät versiot asiakkaan toimipisteiden tietokannoista. Keskitetty varmuuskopiointisovellus päädyttiin tuottamaan itse valmiiden ratkaisujen etsimisen jälkeen, sillä sopivaa vaatimukset täyttävää valmista ratkaisua ei löytynyt.

4.5.1 Vaatimukset

Sovellukselle asetettiin seuraavat vaatimukset:

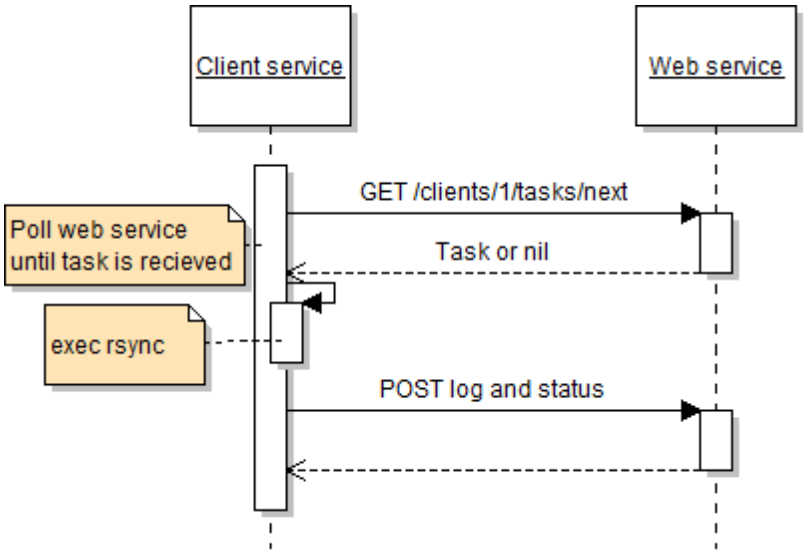
1. Ratkaisun täytyy pystyä siirtämään automaattisesti varmuuskopioita etäpalvelimelle salatun yhteyden ylitse.
2. Varmuuskopiointioperaatiot tulee pystyä ajastamaan kertaluontoisesti, sekä toistuvasti tietyllä aikavälillä.
3. Varmuuskopioita tulee voida hallita keskitetysti yhdeltä palvelinkoneelta konfiguraatiotiedostojen tai käyttöliittymän avulla, siten että käyttäjän ei tarvitse ottaa etäyhteyttä varmuuskopioitaville koneille.
4. Ratkaisun täytyy pystyä autentikoitumaan Windows-levyjaoille, joilla varmuuskopioitavat tietokannat sijaitsevat.
5. Varmuuskopioinnin onnistuminen ei saa riippua sisään kirjautuneesta käyttäjästä tai siitä, onko käyttäjä ylipäänsä kirjautunut sisään.
6. Ratkaisun tulee minimoida tiedonsiirto varmuuskopioinnin kohteen ja varmuuskopiointipalvelimen välillä.

4.5.2 Toteutus

Tiedonsiirto päätettiin toteuttaa hyödyntäen rsync:iä. Rsync on avoimen lähdekoodin ohjelma ja verkko-protokolla, joka minimoi verkon yli siirrettävän datan määrän siirtämällä vain muuttuneet osat tiedostoista.

Sovellus koostuu kahdesta osasta: Ruby + DeltaCopy [17] (rsync kääre Windowsille) -asiakassovelluksesta ja Ruby on Rails + RSync -palvelimesta. Asiakassovelluksen ajoympäristönä toimi Windows XP, Vista tai 7 ja palvelimen ajoympäristönä toimi CentOS Linux-distribuutio.

Perusperiaatteet: palvelimen tietokanta sisältää tiedot varmuuskopiointiasiakkaista ja varmuuskopiointitehtävistä, joiden perusteella se luo varmuuskopiontitehtäväinstansseja. Asiakkaat kyselevät palvelimen web-rajapinnasta tehtäviä esimerkiksi viiden minuutin välein. Kun tehtävä on tarjolla, niin asiakas suorittaa rsync-komennon palvelimelta saaduilla parametreilla ja lähettää sekä komennon tulostaman login että exit-koodin palvelimelle web-rajapinnan kautta (Kuva 4.5).



Kuva 4.5. Varmuuskopiointisovelluksen toimintaperiaate.

Palvelin tarjoaa web-käyttöliittymän varmuuskopioiden ajastamiseen sekä varmuuskopiointitehtävien tilojen ja lokitietojen tarkasteluun.

Asiakas koostuu Ruby-skriptistä ja DeltaCopy rsync-kääreestä. Asiakas-sovellusta ei tarvitse asentaa koneelle, vaan riittää, että siirretään sovellus johonkin hakemistoon ja ajastetaan sen suoritus toistuvaksi käyttäen Windowsin ajastetut tehtävät -toimintoa. Sovelluksen yksi suoritus vastaa siis yhtä tehtävän pollaamista, mikäli palvelin palauttaa tehtävä-instanssin niin se suoritetaan. Ajastetut tehtävät -toiminnolla saadaan asetettua tehtävän suorittavan käyttäjän tunnus ja salasana, jolla on oikeudet lukea lähdesovelluksen tietokantatiedostoja. Näin varmuuskopiointisovelluksen suorittaminen onnistuu riippumatta siitä, kuka on kirjautuneena sisään.

Palvelin on Linux-palvelimelle asennettu Ruby on Rails -sovellus. Se tarjoaa web-käyttöliittymän, jolla voidaan hallita varmuuskopiointiasiakkaita ja varmuuskopiointitehtäviä, sekä tarkastella tehtäväinstanssien tilaa. Varmuuskopioiden ajastus tapahtui luomalla käyttäjän ajastamien tehtävien perusteella cron-tiedosto, jolloin crond huolehtii varmuuskopiointitehtäväinstanssien luomisesta ajallaan.

4.6 Salatun asiatiedon purkusovellus

Salatun asiatiedon purkaminen alkoi tutkimalla sovelluksen ajonaikaista käyttäytymistä CheatEngine -sovelluksen [18] avulla, tavoitteena eristää salauksen purkava aliohjelma. Jonka jälkeen kyseistä aliohjelmaa voitaisiin käyttää salauksen purkamiseen. Lyhyen

tutkimuksen jälkeen kuitenkin huomattiin, että aliohjelman eristäminen on hyvin työlästä ja päätettiin etsiä muita ratkaisuja.

Lopulta salatut tiedot päädyttiin kopioimaan sovelluksen käyttöliittymästä AutoIT-skriptin avulla. AutoIT [19] on käyttöliittymien automaatioon ja skriptaukseen suunniteltu ohjelmointikieli ja työkalusto. Tämä ratkaisu oli helppo toteuttaa, sillä yksi käyttöliittymänäkymä sisälsi kaikki yksittäisen salatun tietokantataulun rivin tiedot ja siihen liittyvien assosiaatioiden pääavainkandidaatit, jolloin salatun tietokantarivin vierasavaimet pystyttiin päättelemään niiden perusteella.

Sovellus koostui kahdesta osasta:

- **AutoIT-skriptistä**, joka kaappasi käyttöliittymästä tiedot XML-tiedostoon ja Ruby-skriptistä, joka jäsensi XML-tiedoston ja lisäsi tietokantarivit MS Access-tiedostoon. AutoIT-skriptejä varten VMWare ESXi-palvelimelle sijoitettiin Windows XP -virtuaalikoneita ajoon, joissa tietojen kaappaus sai edetä rauhassa. Kaappausskriptin ajo yhdelle toimipisteelle kesti noin 6h-48h, riippuen toimipisteen koosta.
- **Ruby-skriptistä**, joka käsitteli AutoIT-skriptin kaappaaman XML-datan, päätteli purettujen rivien vierasavaimet, raportoi virhetilanteista ja suoritti samalla datan puhdistusta poistaen tyhjiä tai duplikaatteja rivejä ja näkymiä. Skripti tallensi datan MS Access-kantaan salattujen tietojen tilalle.

4.7 Automaattinen konversiosovellus

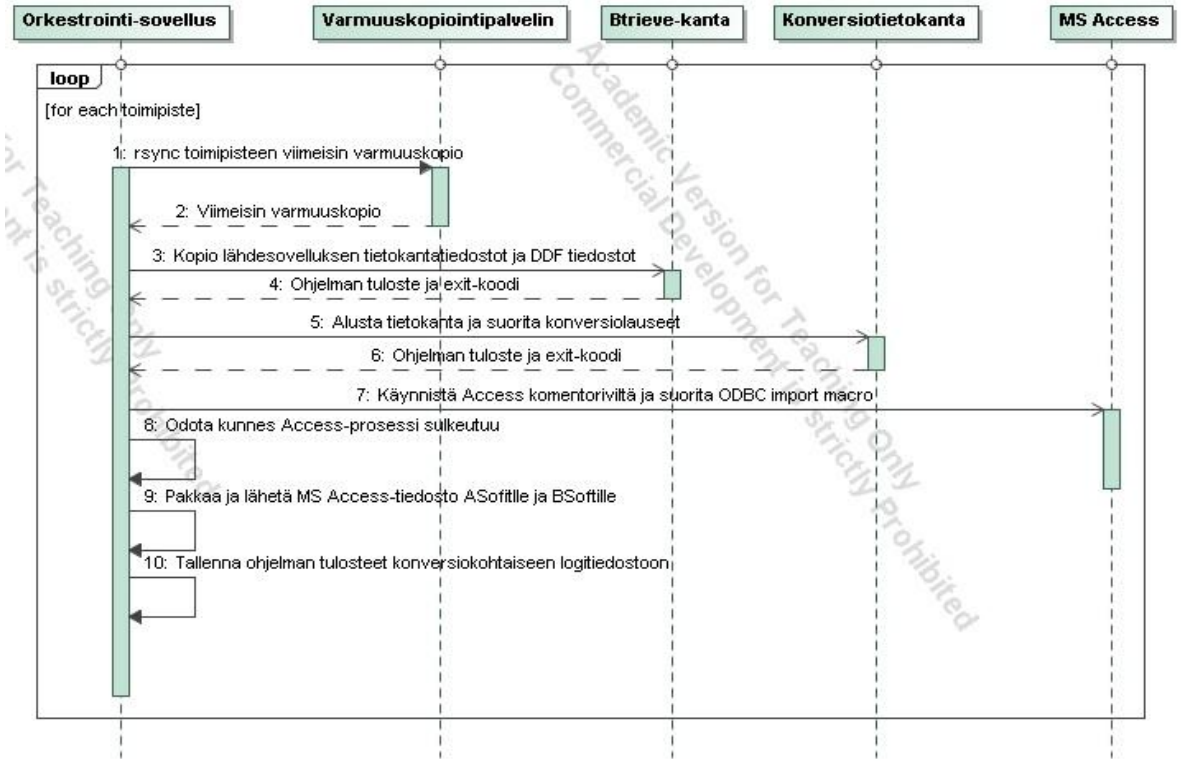
Automaattinen konversiosovellus päätettiin toteuttaa datamigraatioprosessin tueksi, sillä konversiovaiheiden suorittaminen käsin olisi ollut hyvin työlästä kaikille 70 toimipisteelle. Lisäksi sovellus mahdollisti nopean konversioiden iteroinnin ja siten tehokkaamman virheiden havaitsemisen ja korjauksen. Automaattinen konversiosovellus on Ruby-skripti, joka orkestroi konversioon osallistuvien ohjelmistojen suoritusta oikeassa järjestyksessä oikeilla syötteillä. Orkestrointikomponentin suunnittelun suurimpana haasteena oli hallita kaikki aikaisemmin käsin hallitut poikkeamat migraatioprosessissa.

Orkestrointisovellus suoritti konversiosekvenssin (Kuva 4.6) seuraavasti:

1. Konversiosekvenssin alussa orkestrointisovellus suoritti rsync-komennon, joka kopioi konversion kohteena olevan sovellustietokannan viimeisimmän version konvertoitavaksi.
2. Tämän jälkeen orkestrointi-sovellus kopioi Btrieve-tietokantatiedostot ja DDF-builder -työkalulla tuotetut DDF-tiedostot Btrieve-tietokannan datahakemistoon, jolloin konversion kohteena olevan Btrieve-tietokannan tila vastasi lähdesovelluksen tietokannan tilaa.
3. Tämän jälkeen orkestrointi-sovellus suoritti konversion suorittavat SQL-lauseet Pervasive SQL:n komentokehote-sovelluksella ja otti kiinni mahdolliset virheet.
4. Sitten orkestrointi-sovellus suoritti MS Access -ohjelman komentoriviltä, käynnistäen ODBC import -makron. MS Access -prosessin sulkeutumista täytyi

odottaa pollaamalla prosessin tilaa, sillä Access käynnistyy komentoriviltä käynnistettäessä eri shelliin.

5. Sitten orkestrointi-sovellus pakkasi valmiin Access-tiedoston ja lähetti sen ASoftille ja BSoftille.
6. Lopuksi sovellus tallensi konversion lokitiedot konversiokohtaiseen lokiin myöhempää tarkastelua varten.



Kuva 4.6. Automaattisen konversion sekvenssikaavio.

Varsinaisen migraatioprosessin aikana automaattista konversiosovellusta käytettiin ajastamalla sen suoritus Windowsin ajastetut tehtävät -toiminnon avulla yöllä suoritettavaksi. Asiakkaan järjestelmää ei käytetty yön aikana. Konversiosovelluksen suorittamisen jälkeen BSoftin automaattinen transformaatiosovellus vei datan kohdesovellukseen.

4.8 Massakonversio ja uuden järjestelmän testaus

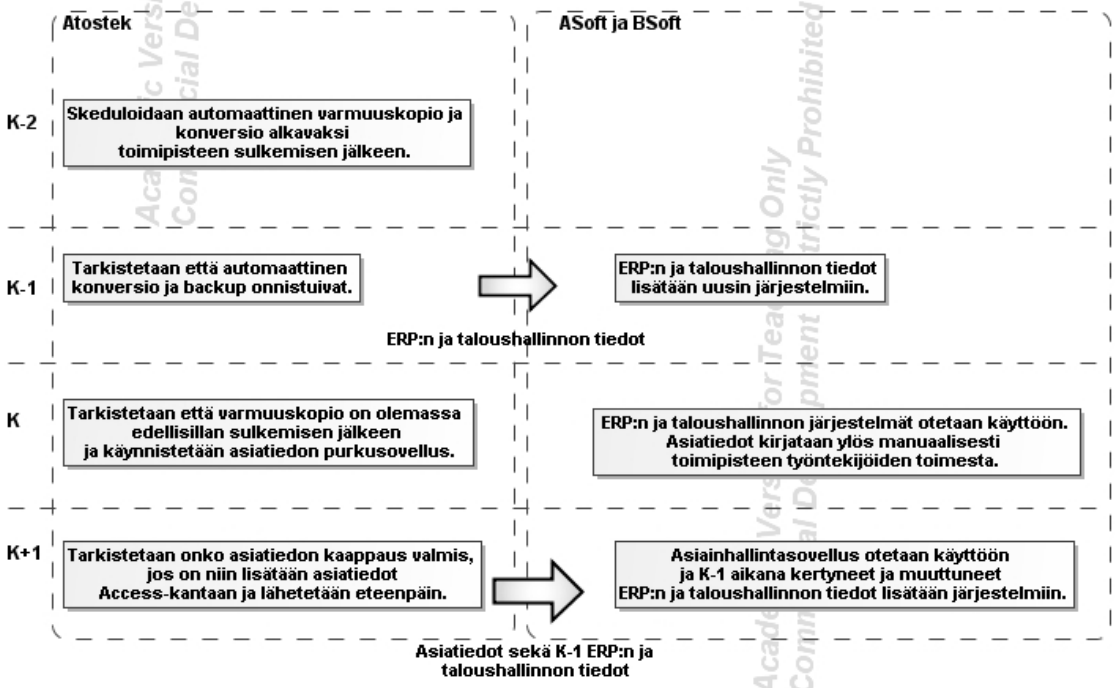
Automaattisen konversiosovelluksen tuottamisen jälkeen ASoftille ja BSoftille toimitettiin kaikkien toimipisteiden konversiot ilman salatun datan purkamista. Vaihe suoritettiin käynnistämällä automaattinen konversioskripti kaikille toimipisteille. Skriptin suorittaminen kesti noin 1,5 vuorokautta. Tämän jälkeen kaikki data ladattiin uusiin järjestelmiin suorituskykytestausta ja datan validointia varten.

Vaihe oli tärkeä suorittaa, koska asiakkaan eri toimipisteissä oli asennettuna lähdesovelluksesta eri versioita. Tällöin oli tärkeää varmistaa, että konversio menee läpi kaikille toimipisteille. Konversioskriptin suorittamisessa ei havaittu ongelmia yhdenkään toimipisteen kohdalla. Mikäli ongelmia olisi havaittu vasta tuotannollisen migraation aikana, korjaus olisi ollut vaikeaa tuotannollisen migraation aikataulun vuoksi.

Uuden järjestelmän testauksessa havaittiin, että datan vieminen kohdesovellukseen hidasti merkittävästi sen käyttöä. Siksi datan vieminen jouduttiin tuotannollisessa migraatiossa suorittamaan sellaisina aikoina, jolloin kohdesovellusta ei käytetä.

4.9 Käyttöönottoprosessi

Käyttöönottoprosessi (tuotannollinen migraatio) suoritettiin toimipiste kerrallaan. Kuvassa 4.7 on esitetty käyttöönottoprosessi. Kuvassa K tarkoittaa käyttöönottopäivää ja K-1 tarkoittaa käyttöönottopäivää edeltävää vuorokautta. Yhdessä vuorokaudessa datamigraation kohteena oli 1-4 toimipistettä.



Kuva 4.7. Käyttöönottoprosessi.

- **Käyttöönottopäivä-2:** Käyttöönottoprosessi alkoi automaattisen varmuuskopiointisovelluksen suorittaman varmuuskopioinnin ajastamisella alkavaksi toimipisteen sulkemisajan jälkeen. Varmuuskopioinnin tiedettiin kestävän maksimissaan 30min, jolloin automaattinen konversiosovellus ajastettiin käynnistymään noin 4h (reilu varmuusvara) varmuuskopioinnin päättymisen jälkeen. Vaiheen tavoitteena oli toimittaa ASoftille ja BSoftille migraation kohteena olevan toimipisteen data yön aikana, jolloin heidän automaattisen transformaatio-ohjelmat veivät datan uuteen järjestelmään. Datamigraatio ei saanut häiritä toimipisteiden päivittäistä tuotannollista toimintaa.
- **Käyttöönottopäivä-1:** Vaiheen tarkoitus oli tarkistaa, että edellisen vaiheen suoritus onnistui. Tämä käytännössä tarkoitti lokitietojen aamuista tarkastelua. Mikäli huomattiin jonkin olevan vialla, niin konversio suoritettiin käsin.
- **Käyttöönottopäivä:** ERP ja taloushallintosovellukset otettiin käyttöön toimipisteissä BSoftin toimesta. Varmistettiin, että toimipisteestä on olemassa varmuus-

kopio K-1 päivän sulkemisen jälkeen, jonka jälkeen ajettiin automaattinen konversiosovellus ja käynnistettiin salatun asiatiedon purkusovellus.

- **Käyttöönottopäivä+1:** Tarkistettiin, onko salatun asiatiedon purkaminen valmistunut. Jos on, niin salattu asiatieto vietiin MS Access-kantaan ja lähetettiin eteenpäin. Asiainhallintasovellus otettiin käyttöön heti kun ASoft vei datan järjestelmään.

Kuva 4.8 on esitettynä ote konversioprosessin aikataulutus-Excelistä. Kuvasta ilmenee kuinka paljon tehtäviä saattoi olla yhtenä konversiopäivänä, joten ehkä hieman rautalangasta väännetty aikataulu todella auttoi suorittamaan kaikki tehtävät ajallaan. Excelin tehtävänä oli myös pitää kirjaa suoritetuista tehtävistä, sillä aina tehtävää ei voitu erilaisista syistä suorittaa ajallaan. Siten oli tärkeää merkitä mihin tilaan tehtävä jäi, jotta se muistettiin suorittaa myöhemmin loppuun.

	Toimipiste	ID	BackupID	ti	13.joulu	Kellonaika	T#	Kuvaus	Suorittaja	Tila
345	Toimipiste A	004	56_115			9:00	4	Varmista että on backup edellisillalta sulkemisen jälkeen. Käynnistä konversio ilman automaattista data-uploadia.	TuM	OK
346	Toimipiste B	176	23_45			9:15	5	Käynnistä salatun asiatiedon konversio.	TuM	OK
347	Toimipiste C	135	71_148			9:00	4	Varmista että on backup edellisillalta sulkemisen jälkeen. Käynnistä konversio ilman automaattista data-uploadia.	TuM	OK
348	Toimipiste D	134	68_139			9:15	5	Käynnistä salatun asiatiedon konversio.	TuM	OK
349						9:00	4	Varmista että on backup edellisillalta sulkemisen jälkeen. Käynnistä konversio ilman automaattista data-uploadia.	TuM	OK
350						9:15	5	Käynnistä salatun asiatiedon konversio.	TuM	OK
351						9:00	4	Varmista että on backup edellisillalta sulkemisen jälkeen. Käynnistä konversio ilman automaattista data-uploadia.	TuM	OK
352						9:15	5	Käynnistä salatun asiatiedon konversio.	TuM	OK
353						9:00	6	Tsekkaa onko salatun asiatiedon XML valmis. Jos on, niin yhdistä XML Access-kantaan ja uploadaa käsin.	TuM	OK
354						9:00	6	Tsekkaa onko salatun asiatiedon XML valmis. Jos on, niin yhdistä XML Access-kantaan ja uploadaa käsin.	TuM	OK
355						9:00	6	Tsekkaa onko salatun asiatiedon XML valmis. Jos on, niin yhdistä XML Access-kantaan ja uploadaa käsin.	TuM	OK
356						9:00	3	Tarkista että automaattibackup ja -konversio onnistuivat. Jos ei, niin tee käsin ja lähetä.	TuM	OK
357						9:00	3	Tarkista että automaattibackup ja -konversio onnistuivat. Jos ei, niin tee käsin ja lähetä.	TuM	OK
358						16:45	1	Yhteys backup-koneeseen: aja backup, tarkista koneen kello, kysy koska sulkevat, skeduloi backup vartti sen jälkeen.	TuM	OK
359						17:00	2	Skeduloi konversio alkamaan puoli tuntia sulkemisen jälkeen => automaattisella data-uploadilla.	TuM	OK
360						16:45	1	Yhteys backup-koneeseen: aja backup, tarkista koneen kello, kysy koska sulkevat, skeduloi backup vartti sen jälkeen.	TuM	OK
361						17:00	2	Skeduloi konversio alkamaan puoli tuntia sulkemisen jälkeen => automaattisella data-uploadilla.	TuM	OK
362						16:45	1	Yhteys backup-koneeseen: aja backup, tarkista koneen kello, kysy koska sulkevat, skeduloi backup vartti sen jälkeen.	TuM	OK
363						17:00	2	Skeduloi konversio alkamaan puoli tuntia sulkemisen jälkeen => automaattisella data-uploadilla.	TuM	OK

Kuva 4.8. Ote konversioprosessin datamigraatiokäsikirjoitus-excelistä yhden päivän osalta.

Prosessilla saatiin minimoitua toimipisteen toiminnalle aiheutunut haitta. Prosessia monimutkaisti salatun tiedon olemassaolo, jonka purkamista ei voitu suorittaa yön ylitse vaiheen hitaudesta johtuen.

4.10 Datan puhdistus

Käyttöönoton päätteeksi jäi jäljelle datan laatuun liittyviä ongelmia joita täytyy ratkaista puhdistamalla dataa. Ongelmia ovat seuraavat:

- Epätäydelliset tietokantarivit: sarakkeita puuttuu.
- Konfliktit samojen asiakkaiden tiedoissa eri toimipisteiden välillä: esimerkiksi asiakkaan henkilötunnus tai sukupuoli saattaa olla eri.
- Asiakkailta puuttuu perustietoja: esimerkiksi henkilötunnus saattaa puuttua. Tällöin asiakas yleensä tunnetaan henkilökohtaisesti pienessä toimipisteessä, jolloin ongelmaa henkilötunnuksen puuttumisesta ei ole ollut. Tällaista dataa ei voi käyttää valtakunnallisesti, sillä esimerkiksi Erkki Lahtisia on suomessa paljon.
- Improvisoitu ohjelman käyttö on tuottanut tietokantaan "roskatietoa". Tämä ei sinällään ole suuri ongelma, vaan pikemminkin kiusa.

Datan puhdistusvaihe on tämän diplomityön kirjoitushetkellä käynnissä, jolloin siihen ei oteta tarkemmin kantaa.

5 ARVIOINTI

Tässä luvussa arvioidaan sekä teoriasta että käytännöstä opittuja asioita. Luvussa asioihin otetaan kantaa aiemmin esitellyn datamigraatioprosessimallin mukaisessa järjestyksessä, sillä kyseinen malli käsittää kuitenkin kaikki datamigraatioprojektiin liittyvät vaiheet korkealla abstraktiotasolla.

5.1 Strategia ja esianalyysi

Strategia ja esianalyysi -vaihe on tärkeä osa datamigraatioprojektia, joka kannattaa suorittaa järjestelmällisesti. Vaihetta voi rinnastaa ohjelmistoprojektin esitutkimusvaiheeseen, jossa huomaamatta jääneiden ongelmien ratkaisemisen työpanos voi kertautua myöhemmissä projektin vaiheissa. Samoin datamigraatioprojektissa työ voi kertautua, jos datamigraatioprosessiin tai datamigraatio-ohjelmiin joudutaan tekemään suuria muutoksia.

5.1.1 Datan sidosryhmien edustajat

Datan sidosryhmien edustajat voidaan selvittää ja dokumentoida alikohdassa 3.2.1 kuvatulla jaottelulla. Sidosryhmien edustajia kuitenkin hyödynnetään projektin jatkuessa ja niiden etsiminen tarpeen tullessa voi hidastaa kyseisen henkilön hallussa olevista tiedoista riippuvan tehtävän suorittamista.

Atostekin toimittamassa projektissa ei jaoteltu henkilöitä teorialuvun kuvaaman datan sidosryhmien edustajien jaottelun mukaisesti. Kolme seuraavaa datan sidosryhmän edustajaa olivat kuitenkin tunnistettavissa projektissa.

- **Tietovaraston omistaja** oli henkilö joka vastasi toimipisteiden ohjelmistojen ja laitteistojen toiminnasta ja hankinnasta.
- **Liiketoiminnan toimialueen asiantuntija** oli sama henkilö kuin tietovaraston omistaja. Hänellä oli käytännön kokemusta toimipisteiden arkipäiväisestä toiminnasta ja lähdejärjestelmän käytöstä. Häneen voitiin olla yhteyksissä sellaisissa kysymyksissä, kuten mitä dataa tähän kenttään kirjoitetaan tai miten jotain lähdejärjestelmän ominaisuutta käytetään.
- **Ohjelmistoasiantuntijoita** olivat käytännössä Atostekin, ASoftin ja BSoftin hankkeeseen osallistuneet työntekijät.
- **Yhtiön data-arkkitehtiä** ei ollut olemassa, mutta tämän tyyppisestä henkilöstä olisi ollut paljon hyötyä projektin kannalta. Häneltä olisi parhaassa tapauksessa saatu lähdejärjestelmän tietokannan dokumentaatio ja skeema, jolloin sitä ei olisi tarvinnut takaisinmallintaa.

- **Teknisen tiedon asiantuntijaa** ei ollut olemassa, mutta tässä projektissa ei välttämättä olisikaan ollut tarpeen, sillä toimipisteiden tietojärjestelmät ja lähdesovellus eivät sisältäneet esimerkiksi monimutkaista käyttäjäoikeuksien hallintaa.

5.1.2 Legacy-tietovarastot

Lähdesovelluksen tietovarastoja oletettiin olevan yksi per toimipiste, mutta projektin jatkuessa paljastui, että joissain toimipisteissä saattoi olla useampia. Tämä johtui yleensä siitä, että kaksi toimipistettä olivat yhdistyneet yhdeksi toimipisteeksi. Lakkautetun toimipisteen tietovarasto oli siirretty jäljelle jääneeseen toimipisteeseen. Tästä ei aiheutunut kuitenkaan viivästyksiä projektin toimittamiseen, koska datamigraatioprosessi vaati vähän käsityötä per toimipiste.

Tässä projektissa legacy-tietovarastojen alikohdan 3.2.2 mukainen tietovarastojen dokumentoiminen olisi todennäköisesti vaatinut enemmän työtä kuin se olisi säästänyt, sillä tietovarastot olivat toimipisteittäin samanlaisia – eri tietovarastotyyppejä oli käytännössä yksi.

Sellaisissa tapauksissa, joissa lähdetietovarastoja on useaa eri tyyppiä voi olla hyväksi dokumentoida kaikki tietovarastotyypit ja sen jälkeen kaikkien tietovarastotyyppien instanssit. Dokumentoimalla tietovarastotyypit ja niiden instanssit erikseen todennäköisesti säästää dokumentaatiotyötä säilyttäen kuitenkin tietovarastojen dokumentoimisesta saavutettavat hyödyt.

5.1.3 Muutettavan datan valitseminen

Muutettavan datan valinnassa ei käytetty formaaleja mallinnusmenetelmiä vaan kohdesovellukseen vietävä tieto dokumentoitiin luomalla MS Access -tietokantaskeema, johon purettava data vietiin. Tämä skeema toimi myös lähtötietona dataa kohdejärjestelmään vievälle ASoftille ja BSoftille.

Tässä projektissa formaaleja mallinnusmenetelmiä ei kaivattu, koska MS Access -tietokantaskeema muodosti riittävän rajapinnan datan tuottajan (Atostek) ja sen käyttäjien (ASoft ja BSoft) välille.

Tietokantaskeema on hyvä tapa dokumentoida tietokannan rakenne, mikäli tietokantataulujen määrä pysyy riittävän vähäisenä. Tällöin tietokannan rakenne on ymmärrettävissä suoraan skeemasta. Tässä vallitsee analogia ohjelmistojen dokumentoinnin kanssa: pienien ohjelmien toiminta ja rakenne voidaan ymmärtää kommentoimattomankin lähdekoodin perusteella, mutta suuremman ohjelmiston ymmärtäminen vaatii avukseen riittävää dokumentointia.

5.1.4 Lähestymistavat

Tässä projektissa käytettiin määritelmän mukaisesti target-pull lähestymistapaa, koska muutettava data valittiin kohdejärjestelmässä tarvitun datan perusteella. Kirjallisuudessa ja artikkeleissa usein esiintyvä target-pull vs. source-push -jaottelu kuulostaa arveluttavalta, sillä uuteen järjestelmään pyritään aina viemään sen tarvitsema data. Kuitenkin

voi olla sellaisia tilanteita, joissa ei tiedetä mitä dataa uudessa järjestelmässä tarvitaan. Tällöin vaihtoehdoksi jää työläämpi source-push -lähestymistapa. Tämän projektin lopussa paljastuikin, että uuteen järjestelmään tarvitaan sellaista dataa, mikä oli tietoisesti jätetty pois valinnoista datan valitsemispalaverissa.

5.1.5 Kuinka siirrytään uuteen järjestelmään

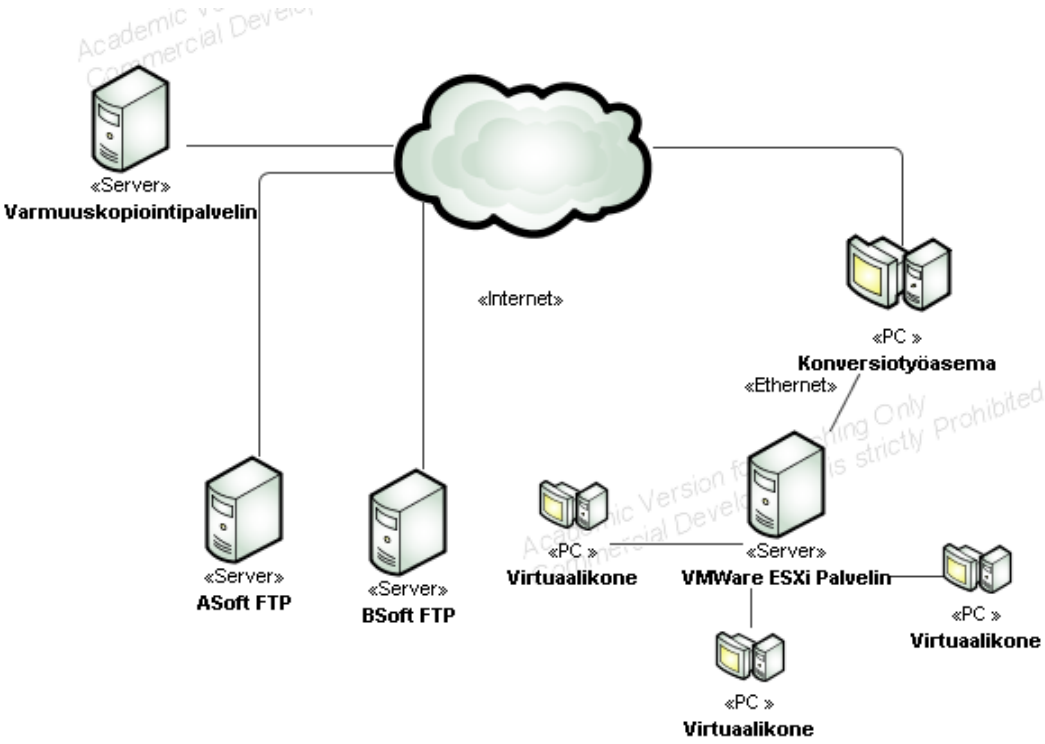
Atostekin toimittamassa projektissa siirtyminen uuteen järjestelmään toteutettiin käytännössä inkrementaalisena migraationa. Tämä tyyli sopi hyvin kyseiseen tapaukseen, jossa dataa muutettiin useasta toisistaan riippumattomasta järjestelmästä yhteen keskitettyyn järjestelmään. Tällöin uuteen järjestelmään voitiin viedä luontevasti yhden toimipisteen datat kerrallaan, jolloin uuden järjestelmän käyttäjäkunta kasvoi tasaista tahtia. Näin migraation liittyvät riskit pysyivät hallinnassa, sillä jokainen toimipiste voitiin käsitellä rauhassa yksilökohtaisesti. Big bang -migraatio olisi ollut käytännössä mahdollista, jos henkilöstö ja tietojärjestelmä resursseja olisi ollut käytettävissä noin 30 kertaisesti.

Aina ei ole mahdollista osittaa lähdejärjestelmää siten, että inkrementaalinen migraatio on mahdollista. Tällöin vaihtoehdoksi jää tehdä big bang -migraatio. Big bang -migraatio vaatii migraatioprosessin ja riskien hallinnan tarkempaa suunnittelua, sillä kerralla uuteen järjestelmään viedään yksinkertaisesti enemmän dataa, jolloin ongelmat ilmaantuvat lyhemmällä aikavälillä.

Rinnakkaista migraatiota kannattaa käyttää kriittisten järjestelmien (päivittäinen liiketoiminta riippuu täysin järjestelmästä) migraatioissa. Tällaisia järjestelmiä voivat olla esimerkiksi verkkokauppa- tai verkkopankkijärjestelmät. Mikäli migraation jälkeen huomataan, että jokin järjestelmän osa ei toimi niin kuin pitäisi, voidaan palata helposti takaisin vanhaan järjestelmään. Tällainen mahdollisuus uuden järjestelmän käyttöönoton jälkeen pienentää huomattavasti pitkien käyttökatkojen todennäköisyyttä.

5.2 Alustan rakentaminen

Tässä projektissa datamigraatioalusta koostui kuvassa 5.1 esitetyistä osista. Varmuuskopiointipalvelin toimi välikätenä lähdesovelluksen datan ja konversiotyöaseman välillä: oli mahdollistettu helppo tiedonsiirto lähdesovelluksesta varmuuskopiointipalvelimelle ja sieltä konversiotyöasemalle. Konversiotyöasema konvertoi automatisoidusti lähdesovelluksen tiedot ASoftille ja BSoftille toimitettavaksi MS Access-tiedostoksi. VMWare ESXi [20] -palvelimella ajettavia virtuaalikoneita käytettiin salatun datan kaappaamiseen lähdesovelluksesta. ASoft FTP ja BSoft FTP toimivat kuvan 2.2 (Datamigraatioarkkitehtuuri) mukaisesti lähdekerääntymisalueina (source staging area). Kuvassa 2.2 esiintyvä datan purkusovellus (data extraction program) ajettiin konversiotyöasemalla ja VMWare ESXi palvelimella ajossa olleilla virtuaalikoneilla.



Kuva 5.1. Datamigraatioalustan infrastruktuuri.

Tässä projektissa ei rakennettu datamigraatioalustaa etukäteen vaan laitteistoa otettiin käyttöön sitä mukaan kuin sitä tarvittiin.

Yleisesti ottaen virtuaalikoneiden hyödyntäminen edesauttaa datamigraatioon liittyvien ohjelmistoympäristöjen ja ohjelmistojen hallintaa ja skaalautuvuutta. Virtuaalikone voidaan palauttaa helposti tiettyyn tilaan, tästä on hyötyä kun datamigraatioon liittyviä ohjelmistoja testataan. Virtuaalikoneesta voidaan myös luoda helposti uusi instanssi, joka on siirrettävissä toiselle laitteistolle. Tällöin datamigraatioon liittyvät ohjelmistot skaalantuvat horisontaalisesti.

5.3 Lähdedatan purkaminen

Lähdedatan purkaminen oli Atostekin varsinainen tehtävä tässä datamigraatioprojektissa. Lähdedatan purkaminen alkoi datanpurkutyökalujen rakentamisella. Tässä datamigraatioprojektissa lähdedatan purkuvaihe oli erityisen haastava, sillä lähdesovelluksen teknistä dokumentaatiota ei ollut saatavilla ja osa datasta oli lisäksi salattua. Tästäkin huolimatta datan purkusovellusten tuottaminen oli kuitenkin vain reilu 25 % projektin kokonaisajankäytöstä.

Suurin ongelma oli lopulta se, kuinka saadaan lähdejärjestelmän data muutettua uuteen järjestelmään ilman, että asiakkaan toimipisteen tuotannollinen toiminta keskeytyy. Tämän vuoksi jouduttiin toteuttamaan varmuuskopiointisovellus ja automaattinen konversioskripti. Lisäksi varmuuskopiointisovelluksen asentaminen kaikkiin asiakkaan toimipisteisiin vei aikaa. Näiden tukisovellusten olemassaolo kuitenkin osoittautui korvaamattomaksi tuotannollisen migraation aikana, sillä ilman niitä tuotannollista migraa-

tiota ei olisi pystytty toimittamaan yhtä tiukalla aikataululla. Lisäksi toistuvien työvaiheiden automatisointi mahdollisimman pitkälle vähentää merkittävästi inhimillisten virheiden määrää.

Kokonaiskuvaa ajatellen datan purkuvaihe on erityisen kriittinen osa datamigraatioprosessia, koska se on datamigraatioprosessin ensimmäinen vaihe, josta muut vaiheet riippuvat. Erityisesti tässä projektissa datan purkamisen epäonnistuminen olisi tarkoittanut tuotannollisen migraatiotyön keskeytymistä myös kahdessa muussa yrityksessä.

5.4 Lähdedatan analyysi

Lähdedatan analyysin suorittivat ASoft ja BSoft, joten tässä diplomityössä tätä työvaihetta ei arvioida toimitetun datamigraatioprojektin kannalta.

Lähdedatan laadun analyysi on tärkeä vaihe, sillä selvitetään onko lähdedatan laatu riittävää kohdejärjestelmän uusille käyttötapauksille. Kohdejärjestelmän käyttötapaukset saattavat vaatia tietynlaatuista dataa toimiakseen. Esimerkiksi vanha järjestelmä ei tee tarkistuksia henkilötunnukselle ja uusi järjestelmä vaatii henkilötunnuksen olevan oikein. Tällöin uusi järjestelmä myös olettaa henkilötunnuksen olevan oikein.

5.5 Lähdedatan puhdistaminen

Lähdedatan puhdistaminen jäi suoritettavaksi kohdetietokannassa ASoftin ja BSoftin toimesta. Datan puhdistamisesta kohdejärjestelmässä on aiheutunut pieniä käytettävyysongelmia kohdejärjestelmään. Lähdedatan puhdistaminen jo lähdetietokannassa olisi mahdollisesti estänyt näiden ongelmien syntymisen.

Lähdedatan puhdistaminen lähdetietokannoissa olisi kuitenkin ollut hyvin monimutkaista tässä projektissa, sillä lähdetietokantoja oli 70, jotka sijaitsivat maantieteellisesti ympäri valtakuntaa. Työ olisi täytynyt suorittaa toimipisteen sulkeutumisen jälkeen tai ennen sen aukaisemista, sillä tämän projektin lähdejärjestelmää ei voida käyttää datan puhdistamisen aikana.

Lähdedatan puhdistamisen sivuuttaminen kokonaan on tietoista ongelmien siirtämistä tulevaisuuteen. Joskus tietysti joudutaan toimimaan näin, mikäli datan puhdistamiselle ei ole käytössä riittäviä resursseja tai se ei syystä tai toisesta ole mahdollista. Tällöin ongelmat kannattaa kuitenkin dokumentoida, jotta niihin voidaan puuttua tulevaisuudessa.

5.6 Datan transformaatio

Datan transformaation tässä projektissa suorittivat ASoft ja BSoft, jolloin sen suoritukseen ei oteta kantaa tässä diplomityössä.

Kirjallisuudessa esitetään, että datan transformaatiosäännöt kannattaa jaotella liiketoiminnalliseen tasoon ja tekniseen tasoon. Tämä jaottelu on sikäli hyvä, että liiketoimintatason transformaatiosäännöillä saadaan mallinnettua kuinka reaalimaailman ilmiö mallinnetaan lähdejärjestelmässä ja kohdejärjestelmässä. Tekninen transformaatiosääntö

puolestaan mallintaa datan tekniset muutokset lähdejärjestelmästä kohdejärjestelmään. Liiketoimintakonseptitason mallinnukseen on olemassa muitakin tekniikoita kuin liiketoimintakonsepti-käsite. Yleisesti tärkeintä on rakentaa jokin konseptitason malli yhteisen terminologian ja keskustelun tueksi. Datan transformaatio -vaiheessa voidaan käyttää hyväksi jo aiemmin luotuja liiketoiminnallisia malleja teknisten transformaatiosääntöjen luomisen apuna.

Tekniset transformaatiosäännöt täytyy määritellä formaalisti, jotta ne olisivat käyttökelpoisia. Formaalin määrittelyn voi tehdä tekstuaalisella kuvauksella tai käyttäen hyväksi jotain mallinnusmenetelmää, mutta myös transformaatio-ohjelma on formaali määrittely tekniselle transformaatiosäännölle. Teknisten transformaatiosääntöjen määrittely ainoastaan ohjelmakoodina on tietysti vähiten työtä vaativa tapa, koska transformaatiosääntö toimii tällöin myös transformaatio-ohjelman osana. Mikäli lisäksi määritellään tekniset transformaatiosäännöt muunlaisina kuvauksina, voidaan tietysti saavuttaa sääntöjen parempi ymmärrettävyys.

5.7 Datan validointi

Tässä projektissa datan validoinnin datamigraatioprosessimallin mukaisesti suorittivat ASoft ja BSoft. Atostekilla varmistettiin, että data on purettu lähdejärjestelmästä oikein. ASoftilla ja BSoftilla varmistettiin, että purettu data on viety kohdejärjestelmään oikein. Datan purkusovellus generoi paljon lokitietoja datan purkamisesta, jolloin virheiden syitä etsittäessä oli helppo varmistaa, että ainakin datan purkaminen oli onnistunut oikein.

Datan validointi on tärkeä osa ennen tuotannollisen migraation aloittamista. Sen tekemättä jättäminen on sama kuin testaisi ohjelmistoa tuotannossa. Ero on se, että datamigraatio-ohjelmia käytetään tuotannossa yleensä vain kerran ja niiden läpi kulkee jos ei kaikki niin ainakin paljon asiakasyrityksen arvokasta liiketoimintadataa, jolloin migraatiossa rikkoutuneen datan pääseminen kohdejärjestelmään voi aiheuttaa huomattavia taloudellisia menetyksiä tai jopa vaaraa ihmisten terveydelle. Tästä syystä datamigraatio-ohjelmien testaaminen on vielä tärkeämpää kuin tavallisten ohjelmien testaaminen. Toisaalta datamigraatio-ohjelmien testaaminen on helpompaa: tarvitsee vain validoida, että kohdejärjestelmään viety data vastaa sisällöltään lähdejärjestelmän dataa.

5.8 Datamigraatioprosessin testaaminen

Datamigraatioprosessin testaamisella varmistetaan, että datamigraatio saadaan suoritettua tietyn aikaikkunan sisällä. Tässä projektissa datamigraatioprosessi testattiin testaamalla yksittäisten vaiheiden kesto ja summaamalla niistä koko prosessin kesto. Tavoite datamigraation ensimmäiselle vaiheelle (salaamattoman datan migraatio) oli saada datat kohdejärjestelmään yön ylitse. Migraation ensimmäinen vaihe ei kuitenkaan kestänyt yhdelle järjestelmälle kuin noin kaksi tuntia, jolloin tehokkuudesta ei tullut ongelmaa.

F. Matthesin [3] mukaan datamigraatioprosessin testaamiseen liittyy myös prosessin dokumentointi. Tämä piti paikkaansa tässä projektissa, sillä datamigraatioprosessi dokumentointiin datamigraatiokäsikirjoitus-exceliin, josta ilmeni kaikille 70 toimipisteelle jokainen datamigraatioprosessin vaihe. Datamigraatioprosessin yksityiskohtainen dokumentoiminen helpotti merkittävästi datamigraation vaiheiden tuotannollista suorittamista oikeaan aikaan ja oikeassa järjestyksessä.

5.9 Kohdesovelluksen testaaminen

Kohdesovellusta testattiin ottamalla se pilottikäyttöön kolmessa toimipisteessä sekä tekemällä kaikkien toimipisteiden datoille massakonversio. Kohdejärjestelmän toiminta on ASoftin ja BSoftin vastuulla, jolloin tässä diplomityössä ei oteta tarkemmin kantaa kyseisen järjestelmän testaukseen.

Ohjelmistojen testaus ei sinällään ole pelkästään datamigraatioon liittyvä ongelma-alue. Datamigraatioprosessimallissa halutaan vain korostaa testaamisen tärkeyttä ennen kuin uusi järjestelmä otetaan tuotantokäyttöön.

5.10 Integraatiotestit ja lopullinen harjoitus

Kohdesovellusten integraatiotestit suorittivat ASoft ja BSoft, joten niihin ei oteta kantaa tässä diplomityössä.

Lopulliset harjoitukset suoritettiin ottamalla uusi järjestelmä käyttöön kolmessa pilottitoimipisteessä. Näin saatiin varmistettua datamigraatioprosessin ja uuden järjestelmän tarkoituksenmukainen toiminta tuotantoympäristössä.

Datamigraatioprosessin harjoittelu on tärkeää sillä ainakin tässä projektissa datamigraatioprosessiin liittyi useita Atostekin, ASoftin ja BSoftin peräkkäisesti suorittamia tehtäviä, joista osa suoritettiin automaattisesti ja osa käsin. Vaiheiden suorituksen tuli myös kestää yhteensä alle 12h. Vaikka suunniteltu datamigraatiomigraatioprosessi sujuikin hyvin kerta heitolla, olisi mahdollisten ongelmien korjaaminen ollut vaikeaa tuotannollisen migraation aikana.

F. Matthesin mainitsema datamigraatiokäsikirjoitus-termi kuvaa hyvin tämän vaiheen lopputulosta, sillä datamigraatioprosessin harjoittelun jälkeen käytössä oli dokumentoitu ja harjoiteltu sarja tehtäviä, jotka kattoivat koko datamigraatioprosessin. Lisäksi datamigraatioprosessin suorittamiseen osallistuvilla henkilöillä oli käytännön kokemusta migraatioprosessin suorittamisesta, mikä tietysti osaltaan edesauttoi tuotannollisen migraation suorittamista aikataulussa.

5.11 Tuotannollinen migraatio

Tässä projektissa tuotannollinen migraatio oli pitkä prosessi, joka kesti kaksi kuukautta. Tuotannollinen migraatio suoritettiin projektissa luodun datamigraatioprosessin perusteella onnistuneesti. Ainoana yllätyksenä olivat lakkautettujen toimipisteiden löytyneet tietokannat, jotka kuitenkin saatiin sovitettua aikatauluun ongelmitta. Ei paluuta -piste

saavutettiin jokaisen toimipisteen kohdalla käyttöönottopäivänä, vanha järjestelmä jäi saataville kullekin toimipisteelle historiatietojen saannin turvaamiseksi.

Tuotannollinen migraatio oli jaettu selkeästi vaiheittain suoritettavaksi. Vaiheiden suorittamisen valvomista oikeassa järjestyksessä hallittiin käsin, sekä orkestrointisovelluksen avulla.

Atostekin toimittamassa projektissa datamigraatioprosessi kulki nimellä käyttöönottoprosessi, mutta ominaisuuksiltaan se silti vastasi datamigraatioprosessia. Prosessi oli myös dokumentoitu, testattu ja harjoiteltu.

5.12 Projektinhallinta

Datamigraatioprosessimallia ei kannata ajatella kuvaavan datamigraatioprojektin suorittamista kuvaavana jäykkänä mallina vaan pikemminkin datamigraatioprojektin pilkkomisena pienempiin ja hallittavampiin osiin. Datamigraatioprosessimallin vaiheiden huomioonottaminen datamigraatioprosessimallin kuvaamassa järjestyksessä auttaa datamigraatioprojektin suunnittelussa, koska se tarjoaa listan datamigraatiossa taklattavista ongelmista. Datamigraatioprosessimalli ei ole jäykkä vaan siitä voidaan jättää tarvittaessa vaiheita suorittamatta. Kuitenkin jonkin vaiheen suorittamatta jättäminen tulisi tehdä tietoisesti ja hyvin perustein. Esimerkiksi lähdedatan laadun analyysi ja sitä seuraava lähdedatan puhdistaminen saattavat olla vaiheita, joiden suorittamatta jättäminen saattaa tuoda yllätyksellisesti suuria ongelmia myöhemmissä vaiheissa.

5.13 Datamigraatio yrityksen palvelutuotteena

Datamigraatioprojektin suorittaminen eroaa ohjelmistoprojektista usealla eri tavalla, kuten tässä diplomityössä on havaittu. Datamigraatioprojektiin liittyy tietty joukko ongelmia, jotka täytyy ottaa huomioon optimaalisen lopputuloksen saavuttamiseksi. Näiden ongelmien järjestelmällinen hallinta on erityistä asiantuntijaosaamista, jota datamigraatiota tarvitseva yritys ei yleensä omaa, koska tietojärjestelmäuudistuksia suoritetaan yrityksissä harvoin. Datamigraatiopalvelu kannattaa siis hankkia ennemmin datamigraatiopalveluita tarjoavalta asiantuntijayritykseltä.

Datamigraatio on J. Morrisin [1] mukaan enemmän liiketoiminnallisen ja liiketoiminnallisen osaamisen ongelma kuin teknisen osaamisen. Tämä pitää sikäli paikkaansa, että datamigraatioasiantuntijapalveluun kuuluu konkreettisen datamigraation suorittavien ohjelmien tuottamisen lisäksi datamigraatioprosessin suunnittelua ja muutettavan datan valitsemista. Datamigraatioprosessin suoritustapa ja lähdedatan valitseminen vaikuttavat asiakkaan päivittäiseen liiketoimintaan. Strategia- ja esianalyysivaiheeseen voi kuulua myös kohdejärjestelmän määrittelyä, jossa onnistuminen vaatii ohjelmistotuotannon menetelmien lisäksi asiakkaan liiketoiminnan ymmärtämistä.

Teknisen osaamisen tarvetta ei kuitenkaan voida täysin sivuuttaa vaan lähdedatan purku, datan transformaatio tai datan vienti kohdejärjestelmään voivat olla teknisesti haastavia. Lisäksi datamigraatiota tukevien ohjelmistojen tuottaminen tai käyttöönotto

vaatii teknistä osaamista. Näiden ohjelmistojen olemassaolo edistää projektin aikatauluja budjettitavoitteiden saavuttamista. Datamigraatiota tarjoavan yrityksen kompetenssiin siis tulisi kuulua sekä konsultaatio-osaamista että teknistä osaamista. Lisäksi valmiit migraatioprosessia tukevat sovellukset ovat myös eduksi.

Datamigraatioprojektia tarjottaessa voi olla kuitenkin vaikeaa perustella tiettyjä datamigraatioon liittyviä töitä, jotka saattavat asiakkaasta tuntua lisätöiltä. Näitä töitä ovat esimerkiksi datan laadun analyysi ja lähdedatan puhdistus. Asiakkaasta lisätöiltä tuntuvia töitä tarjottaessa täytyy voida perustella asiakkaalle datan laadun paranemisesta saatava hyöty päivittäiselle liiketoiminnalle. Lisäksi asiakkaalle voidaan sanoa, ettei uusi järjestelmä tule vastaamaan hänen odotuksiaan, mikäli siihen viedään vanhasta järjestelmästä erittäin heikkolaatuista dataa.

6 YHTEENVETO

Tämä diplomityö esitteli F. Matthesin [3] koostaman datamigraatioprosessimallin vaiheet sekä Atostekin suorittaman datamigraatioprojektin. Työssä havaittiin, että datamigraatioprojektissa on tärkeää rakentaa datamigraatioprosessi, jonka avulla tuotannollinen migraatio suoritetaan. F. Matthesin datamigraatioprosessimallia voidaan käyttää datamigraatioprosessin rakentamisen apuna. Datamigraatioprosessimalli ei kuitenkaan tarjoa universaalia ratkaisua datamigraatio-ongelmaan vaan kehikon, jonka ympärille ratkaisu voidaan rakentaa.

Atostekin toimittamaa datamigraatioprojektia ei suunniteltu datamigraatioihin liittyvien prosessimallien perusteella. Projektin läpiviennissä oli kuitenkin tunnistettavissa datamigraatioprosessimallin mukaiset vaiheet (Taulukko 6.1). Projektissa luotiin testattu ja harjoiteltu datamigraatioprosessi, jonka avulla datamigraatio suoritettiin tuotannossa. Luotu prosessi soveltuu monitoimipistejärjestelmän datamigraatioon keskitettyyn tietokantaan. Prosessia ja siihen liittyivä tukisovelluksia voidaan uudelleenkäyttää vastaavanlaisissa datamigraatioprojekteissa.

Taulukko 6.1. Datamigraatioprosessimallin vaiheet Atostekin toimittamassa projektissa.

Datamigraatioprosessimalli	**Atostekin projekti**
Strategia ja esianalyysi.	Esitutkimusvaihe.
Datamigraatioalustan pystyttäminen.	Datamigraatioalusta kehittyi projektin aikana.
Lähdedatan purku.	Lähdedatan purku oli Atostekin vastuulla. Lähdedata toimitettiin MSAccess-tietokannoissa ASoftille ja BSoftille.
Lähdedatan analyysi.	Lähdedatan analysoinnin suorittivat ASoft ja BSoft.
Lähdedatan puhdistaminen.	Lähdedatan puhdistamista suoritetaan parhaillaan tätä työtä kirjoitettaessa.
Datan transformaatio.	ASoft ja BSoft suorittivat datan transformaation omiin järjestelmiinsä.
Datan validaatio.	Datan purkuvaiheessa varmistettiin, että data on purettu oikein. Datan validaatiota transformaation jälkeen suorittivat ASoft ja BSoft.

Datamigraatioprosessi-testit.	Datamigraatioprosessi testattiin testaamalla jokainen vaihe erikseen ja summaamalla niistä datamigraatioprosessin suorituksen kesto.
Kohdesovelluksen testaus.	Kohdesovelluksen testauksen suorittivat ASoft ja BSoft.
Integraatiotestit ja lopullinen harjoitus.	Integraatiotestit suorittivat ASoft ja BSoft. Loppullisessa harjoituksessa uusi järjestelmä otettiin pilottikäyttöön kolmessa toimipisteessä.
Tuotannollinen migraatio ja viimeistely.	Tuotannollinen migraatio suoritettiin inkrementaalisesti toimipiste kerrallaan.

Datamigraation kertaluontoisuudesta seuraa, että datamigraation aikana luotuja datan purku, -transformaatio ja -vientisovelluksia käytetään yleensä tuotannossa vain kerran. Täten niiden ylläpidosta datamigraation jälkeen ei tarvitse huolehtia. Tällöin heikosti ylläpidettävien, mutta nopeiden toteutustekniikoiden käyttö datamigraatioprojekteissa on sallittua. Heikko ylläpidettävyys ei kuitenkaan saa haitata datamigraatio-ohjelmistojen kehitystä datamigraatioprojektin aikana.

Datamigraatioprosessi muotoutuu asiakkaan liiketoiminnan ja asiakkaan vanhan järjestelmän määrittämien reunaehtojen ympärille. Yleensä asiakkaan päivittäinen liiketoiminta riippuu datamigraation kohteena olevasta järjestelmästä, jolloin datamigraatio täytyy suorittaa tietynpituisen suunnitelmallisen käyttökatkon eli datamigraatioikkunan aikana. Datamigraatioikkunan pituudesta riippuu, voidaanko migraatio suorittaa kerralla vai pitääkö datamigraation kohteena oleva data osittaa pienempiin kokonaisuuksiin. Ositetut kokonaisuudet viedään uuteen järjestelmään usean lyhyemmän käyttökatkon aikana. Eräs mahdollisuus on implementoida datan synkronointi uuden ja vanhan järjestelmän välille, jolloin uutta ja vanhaa järjestelmää voidaan käyttää rinnakkain. Tällöin myös saavutetaan mahdollisuus palata vanhaan järjestelmään tarvittaessa.

Datamigraatioprosessin suoritusnopeutta arvioidessa on hyvä huomata, että datamigraatioprosessin suoritusnopeus koostuu sekä ihmisten että ohjelmistojen tekemistä asioista. Siten datamigraatioprosessin suoritusnopeus saadaan selville ainoastaan kuivaharjoittelemalla datamigraatioprosessin suorittaminen. Datamigraatioprosessin harjoittelun perusteella tuotetaan myös datamigraatiokäsikirjoitus, josta ilmenee kaikki datamigraatioprosessiin liittyvät työtehtävät ja niiden suoritusajat. Tuotannollisen datamigraatioprosessin yksityiskohtainen kuvaus auttaa havaintojen mukaan datamigraation tuotannollista suorittamista. Edelleen tehtävien suorittamisesta kerätyt lokitiedot tulevat tarpeen virheitä metsästettäessä, jolloin huolellinen lokitietojen tuottaminen kannattaa.

Useassa lähteessä esitelty datamigraatioalustan rakenne esiintyy kaikissa datamigraatioprojekteissa, sillä se on määritelty tarpeeksi korkealla abstraktiotasolla. Datamigraatioihin väistämättä liittyy aina datan purkusovellus, transformaatiosovellus ja datan vientisovellus. Lähdekerääntymisalue tarvitaan datan eriyttämiseksi lähdejärjestelmästä. Kohdekerääntymisalue tarvitaan varastoimaan transformoitua dataa vietäväksi uuteen

järjestelmään. Orkestrointikomponentin tehtävänä on hallita datamigraatioon liittyvien sovellusten suoritusta oikeassa järjestyksessä ja haluttuna ajankohtana. Orkestrointikomponentti voi myös tukea datamigraation suorituskyvyn skaalautuvuutta laitteistokapasiteetin mukaan.

Datamigraatiossa täytyy myös parantaa datan laatua kohdejärjestelmän vaatimusten mukaiseksi. Asiakkaan motivaatio järjestelmäuudistukselle on usein parantaa järjestelmän laatua. Tämä ei onnistu, jos uuteen järjestelmään viedään vanhan järjestelmän heikkolaatuinen data jalostamattomana. Datan laadun analyysin tarkoituksena on selvittää onko datan laatu riittävää uuteen järjestelmään.

Datamigraatio yleensä seuraa tietojärjestelmäuudistuksista, joita tehdään niiden kalleudesta ja riskeistä johtuen harvoin. Näin datamigraatio-osaamista ei pääse kertymään vain datamigraatiota omien tietojärjestelmien uudistusten yhteydessä suorittavaan yritykseen. Tästä seuraa, että datamigraatio on oivallinen palvelutuote. Datamigraatiopalvelua tarjoavalla asiantuntijayritykselle kertyy kokemusta datamigraatioprojekteista, jota datamigraatiota tarvitsevan osapuolen kannattaa ennemmin ostaa kuin kehittää omaa.

Järjestelmäuudistuksiin puolestaan ajaudutaan yrityksissä väistämättä tässä diplomityössä todetuista syistä, jolloin datamigraatiot projektityyppinä tulevat tuskin katoamaan maailmasta. Palveluarkkitehtuureihin perustuvien järjestelmien yleistyminen kuitenkin voi vähentää datamigraatioiden määrää tulevaisuudessa. Palveluarkkitehtuureissa datavarastot suunnitellaan monikäyttöisiksi, jolloin uusien ominaisuuksien tuominen järjestelmään vaatii harvemmin datavarastojen tietorakenteiden muokkaamista.

7 LÄHDELUETTELO

1. **Morris, John.** *Practical Data Migration.* Swindon : British Computer Society, 2006. 1-902505-71-9.

2. **Mallory, Cheri.** What Do I Mean When I Say Data Migration? *Informatica - Data integration company.* [Online] Informatica, 11. Helmikuu 2010. [Viitattu: 2012. Tammikuu 27.] http://blogs.informatica.com/perspectives/index.php/2010/02/11/what-do-i-mean-when-isay-data-migration/.

3. **Florian Matthes, Christopher Schulz.** Towards an integrated data migration process model. *Software Engineering for Business Information Systems (sebis).* [Online] Huhtikuu 2011. [Viitattu: 27. Tammikuu 2012.] http://wwwmatthes.in.tum.de/file/attachments/wikis/sebis-article-archive/ms11-towards-an-integrated-data-migration/tb_DataMigration.pdf.

4. EUROOPAN PARLAMENTIN JA NEUVOSTON DIREKTIIVI 2007/2/EY, annettu 14 päivänä maaliskuuta 2007, Euroopan yhteisön paikkatietoinfrastruktuurin (INSPIRE) perustamisesta. *DIREKTIIVIT.* [Online] 14. Maaliskuu 2007. [Viitattu: 30. Tammikuu 2012.] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2007:108:0001:0014:FI:PDF.

5. **Harsu, Maarit.** *Ohjelmien ylläpito ja uudistaminen.* Tampere : Talentum Media Oy, 2003. 951-762-829-3.

6. *Testing & Quality Assurance in Data Migration Projects.* **Florian Matthes, Christopher Shulz, Klaus Haller.** s.l. : 27th IEEE International Conference on Software Maintenance, 2011. 1063-6773.

7. **Haller, Klaus.** Towards the Industrialization of Data Migration: Concepts and Patterns for Standard Software Implementation Projects. *Klaus Haller's Homepage.* [Online] 12. Kesäkuu 2009. [Viitattu: 27. Tammikuu 2012.] http://www.klaushaller.net/media/haller_datamigration_caise2009.pdf.

8. **Bloor Research.** Data Migration Survey. *Independent IT Research and Analysis | Bloor.* [Online] Syyskuu 2007. [Viitattu: 6. Helmikuu 2012.] http://www.bloorresearch.com/download/freepaper/876.html.

9. **Ambler, Scott W.** 2010 IT Project Success Rates Survey Results. [Online] Ambysoft, Kesäkuu 2010. [Viitattu: 30. Tammikuu 2012.] http://www.ambysoft.com/surveys/success2010.html.

10. The essential online community resource for the data migration profession. *The data migration go-live strategy - what is it and why does it matter?* [Online] Data Migration Pro, 26. Maaliskuu 2009. [Viitattu: 3. Helmikuu 2012.] http://www.datamigrationpro.com/data-migration-articles/2009/3/26/the-data-migration-go-live-strategy-what-is-it-and-why-does.html.

11. **Informatica.** Informatica. *Informatica - Data Quality.* [Online] Informatica. [Viitattu: 6. Helmikuu 2012.] http://www.informatica.com/us/products/data-quality/data-quality/.

12. **Zamzami, I.F.;Fatani, H.A.A. ja Zammarah, N.A.H.** *Data migration challenges: The impact of data quality — Case study of University Putra Malaysia UPM.* Kuala Lumpur : Research and Innovation in Information Systems (ICRIIS), 2011 International Conference, 2011. 978-1-61284-295-0.

13. **Shubho, Al-Farooque.** Understanding Set based and Procedural approaches. *Codeproject.* [Online] 15. Maaliskuu 2009. [Viitattu: 13. Helmikuu 2012.] http://www.codeproject.com/Articles/34142/Understanding-Set-based-and-Procedural-approaches.

14. **Staab, Tom.** Set-Based vs. Procedural. *ASK SQL.* [Online] 28. Lokakuu 2009. [Viitattu: 13. Helmikuu 2012.] http://ask.sqlservercentral.com/questions/2019/set-based-vs-procedural.html.

15. *ChunkVNC.* [Online] [Viitattu: 2012. Helmikuu 27.] http://www.chunkvnc.com/.

16. *UltraVNC.* [Online] [Viitattu: 29. Maaliskuu 2012.] http://www.uvnc.com/.

17. *Deltacopy - Rsync for Windows.* [Online] Synametrics Technologies Inc. [Viitattu: 29. Maaliskuu 2012.] http://www.aboutmyip.com/AboutMyXApp/DeltaCopy.jsp.

18. *CheatEngine.* [Online] [Viitattu: 3. Maaliskuu 2012.] http://www.cheatengine.org/.

19. *AutoIT - Automation and Scripting Language.* [Online] [Viitattu: 3. Maaliskuu 2012.] http://www.autoitscript.com/.

20. VMWare ESXi. *VMWare.* [Online] VMWare, 2012. [Viitattu: 10. Toukokuu 2012.] http://www.vmware.com/products/vsphere/esxi-and-esx/index.html.

21. *Push to Pull: How Lean Concepts Improve a Data Migration.* **Bradley, Rand.** 2007. 0-7695-2872-4 .

22. **Potter, Philip Howard & Carl.** Data Migration in the Global 2000: Research, forecasts and survey results. [Online] 2007. [Viitattu: 27. Tammikuu 2012.] http://www.rever.eu/white-papers/876-Data-Migration-Survey.pdf.

23. **Informatica.** 5 Pitfalls of Data Migration. *Scridb.* [Online] Heinäkuu 2010. [Viitattu: 6. Helmikuu 2012.] http://www.scribd.com/doc/53626934/5-Pitfalls-of-Data-Migration.

24. *DeltaCopy.* [Online] [Viitattu: 2. Maaliskuu 2012.] http://www.aboutmyip.com/AboutMyXApp/DeltaCopy.jsp.

25. *Spring Batch.* [Online] 21. Marraskuu 2011. [Viitattu: 19. Huhtikuu 2012.] http://static.springsource.org/spring-batch/index.html.